ioProgrammo Anno XI - N°11 (120)

ANTEPRIMA: MICROSOFT ENTITY FRAMEWORK
IL PRODOTTO PER GESTIRE IN MODO UNIFORME OGNI TIPO DI DATABASE

VERSIONE PLUS RIVISTA+LIBRO+CD €9,90 — VERSIONE STANDARD RIVISTA+CD €6,90

ioPROGRAMMO

PER ESPERTI E PRINCIPIANTI

Poste Italiane S.p.A Spedizione in A.P. • D.L. 353/2003 (conv.in L.27/02/2004 n.46) art.1 comma 2 DCB ROMA Periodicità mensile • NOVEMBRE 2007 • ANNO XI, N.11 (120)

ESTENDI IL TUO BLOG

Ecco come usare le API di WordPress per creare plugin personalizzati

- **ANATOMIA DI UN ADDON** Crea uno scheletro vuoto e rendilo visibile nel pannello di amministrazione
- **HOOK E FILTRI** Intercetta gli eventi che accadono sul sito e reagisci di conseguenza
- **GESTIONE** Aggiungi una pagina nell'interfaccia privata e usala per configurare le opzioni dell'addon
- **ESEMPIO PRATICO** Ecco un plugin che si connette ad un account di Gmail e pubblica sul blog le email flaggate con una star

INSERTO SPECIALE

La miniguida ai linguaggi da imparare per lavorare come programmatore
Scopri qual è il linguaggio che fa per te ed in quale campo applicarlo per avere un curriculum di successo!

.NET
ELABORA I TUOI GRAFICI
Vuoi ottenere report visuali dai tuoi dati? Vuoi visualizzarli in Excel o sul Web? Noi ti diciamo come fare

PHP
OGGETTI E CLASSI QUESTI SCONOSCIUTI
Ti spieghiamo passo dopo passo quali sono le funzionalità più potenti e meno sfruttate di PHP

XML
XPATH IL TROVATUTTO
Usa il tuo file XML come un database e trova velocemente i contenuti tramite una query di ricerca

ACTIONSCRIPT
INTRODUZIONE A FLEX 2.0
Lascia disegnare i grafici! Tu scrivi il codice XML, compilalo ed ottieni una completa interfaccia grafica

RI-PROGRAMMA IL TUO CELLULARE

Intercetta una chiamata, un sms, o qualunque altro evento e gestiscilo come vuoi tu. Ad esempio stacca il Bluetooth se la carica della batteria è troppo bassa...

ASP.NET
PROTEGGITI DALLO SPAM CON IL CAPTCHA
Genera caratteri casuali e confrontali con quelli digitati dall'utente

SQL
SFRUTTA LE PROPRIETÀ ESTESE DI SQL
Usa le note di campi e righe e crea automaticamente le label delle form

JAVA
JAVA ED IL NETWORKING NATIVO
Scrivi un server completo utilizzando il linguaggio di SUN

CORSI BASE
- **WxWidgets** Realizza un paintbrush multipiattaforma
- **PocketPC** Crea un orologio con sfondo dinamico

SOLUZIONI ALGORITMI DI SCHEDULING: DALLA GESTIONE DELLA CPU, ALLA PROGRAMMAZIONE DELLE ATTIVITÀ LAVORATIVE
ECCO COME PIANIFICARE OGNI COSA CON PRECISIONE MATEMATICA

EDIZIONI MASTER
www.edmaster.it

ESTENDI IL TUO BLOG • RIPROGRAMMA IL TUO CELLULARE • PROTEGGITI DALLO SPAM CON IL CAPTCHA • JAVA E IL NETWORKING NATIVO
EDIZIONI MASTER

Anno XI - N.ro 11 (120) - Novembre 2007 - Periodicità: Mensile
Reg. Trib. di CS al n.ro 593 del 11 Febbraio 1997
Cod. ISSN 1128-594X
E-mail: ioprogrammo@edmaster.it
http://www.edmaster.it/ioprogrammo
http://www.ioprogrammo.it

Direttore Editoriale: Massimo Sesti
Direttore Responsabile: Massimo Sesti
Responsabile Editoriale: Gianmarco Bruni
Vice Publisher: Paolo Soldan
Redazione: Fabio Farnesi
Collaboratori: R. Allegra, A. Galeazzi, F. Grimaldi, M. Locuratolo, O. Peli, G. Dattilo, P. Perrotta, A. Pelleriti,
Segreteria di Redazione: Rossana Scarcelli

Realizzazione grafica: Cromatika S.r.l.
Art Director: Paolo Cristiano
Responsabile grafico di progetto: Salvatore Vuono
Coordinamento tecnico: Giancarlo Sicilia
Illustrazioni: M. Veltri
Impaginazione elettronica: Francesco Cospite, Lisa Orrico, Nuccia Marra, Luigi Ferraro

Realizzazione Multimediale: SET S.r.l.
Realizzazione CD-Rom: Paolo Iacona

Pubblicità: Master Advertising s.r.l.
Via C. Correnti, 1 - 20123 Milano
Tel. 02 831212 - Fax 02 83121207
e-mail advertising@edmaster.it
Sales Director: Max Scortegagna
Segreteria Ufficio Vendite: Daisy Zonato

Editore: Edizioni Master S.p.a.
Sede di Milano: Via Ariberto, 24 - 20123 Milano
Sede di Rende: C.da Lecco, zona industriale - 87036 Rende (CS)
Presidente e Amministratore Delegato: Massimo Sesti
Direttore Generale: Massimo Rizzo

ABBONAMENTO E ARRETRATI
ITALIA: Abbonamento Annuale: IOPROGRAMMO (11 NUMERI) €59,90 SCONTO 21% SUL PREZZO DI COPERTINA DI €75,90 · IOPROGRAMMO CON LIBRO (11 NUMERI) €75,90 SCONTO 30% SUL PREZZO DI COPERTINA DI €108,90 OFFERTE VALIDE FINO AL 31/12/07.
Costo arretrati (a copia): il doppio del prezzo di copertina + € 5.32 spese (spedizione con corriere). Prima di inviare i pagamenti, verificare la disponibilità delle copie arretrate allo 02 831212.
La richiesta contenente i Vs. dati anagrafici e il nome della rivista, dovrà essere inviata via fax allo 02 83121206, oppure via posta a EDIZIONI MASTER via C. Correnti, 1 - 20123 Milano, dopo avere effettuato il pagamento, secondo le modalità di seguito elencate:
- cc/p n.16821878 o vaglia postale (inviando copia della ricevuta del versamento insieme alla richiesta);
- assegno bancario non trasferibile (da inviarsi in busta chiusa insieme alla richiesta);
- carta di credito, circuito Visa, Cartasì, o Eurocard/Mastercard (inviando la Vs. autorizzazione, il numero di carta di credito, la data di scadenza, l'intestatario della carta e il codice CVV2, cioè le ultime 3 cifre del codice numerico riportato sul retro della carta).
- bonifico bancario intestato a Edizioni Master S.p.A. c/o BCC MEDIOCRATI S.C.AR.L. c/c 0 000 000 12000 ABI 07062 CAB 80880 CIN P (inviando copia della distinta insieme alla richiesta).

SI PREGA DI UTILIZZARE IL MODULO RICHIESTA ABBONAMENTO POSTO NELLE PAGINE INTERNE DELLA RIVISTA. L'abbonamento verrà attivato sul primo numero utile, successivo alla data della richiesta.
Sostituzioni: qualora nei prodotti fossero rinvenuti difetti o imperfezioni che ne limitassero la fruizione da parte dell'utente, è prevista la sostituzione gratuita, previo invio del materiale difettato.
La sostituzione sarà effettuata se il problema sarà riscontrato e segnalato entro e non oltre 10 giorni dalla data effettiva di acquisto in edicola e nei punti vendita autorizzati, facendo fede il timbro postale di restituzione del materiale.
Inviare il CD-Rom difettoso in busta chiusa a:
Edizioni Master - Servizio Clienti - Via C. Correnti, 1 - 20123 Milano
Assistenza tecnica: ioprogrammo@edmaster.it

Servizio Abbonati:
☎ tel. 02 831212
@ e-mail: servizioabbonati@edmaster.it

Stampa: Arti Grafiche Boccia S.p.a. Via Tiberio Felice, 7 Salerno
Stampa CD-Rom: Neotek S.r.l. - C.da Imperatore - Bisignano (CS)
Distributore esclusivo per l'Italia: Parrini & C S.p.A.
Via Vitorchiano, 81 - Roma

Finito di stampare nel mese di Ottobre 2007

Certificato UNI EN ISO 14001

Questo mese su ioProgrammo

▼ SVILUPPO: C'È UN LIMITE?

Le aziende hanno grandi esigenze. I programmatori inventano metodi nuovi per trovare soluzioni adeguate La richiesta di produttività sempre più elevata inventa problemi complessi a cui rispondere ogni giorno. Le aziende che sviluppano software per programmatori realizzano nuovi e più grandi progetti che aiutino i loro clienti nel trovare soluzioni adeguate alle esigenze delle aziende. Il risultato di questo circolo vizioso è: "sistemi troppo complessi". Nel tentativo di agevolare il processo di sviluppo siamo arrivati ad un punto dove realizzare una qualunque applicazione significa acquisire un certo numero di competenze e interagire con persone e strumenti che talvolta ci sono del tutto sconosciuti. Spesso e volentieri i programmatori si trovano a scrivere pezzi di codice per progetti rispetto ai quali non hanno la minima idea di quale sia lo scopo finale. Sono i così detti progettisti del software a tirare le fila dell'insieme pur non conoscendo, frequentemente, la tecnologia su cui si basa. I sistemisti infine devono fare veri e propri salti mortali per mettere insieme "accrocchi" di applicazioni che non comunicano fra loro. Ora, non siamo sfavorevoli allo sviluppo. Al contrario una rivista come la nostra promuove la tecnologia nel suo insieme e mostra come correttamente utilizzarla per i propri scopi. È questa la via da seguire però: non complicare inutilmente cose semplici. Ogni tecnologia ha un suo campo d'applicazione ed è del tutto inutile utilizzare tecniche complesse su problemi che possono essere risolti con poche righe di codice. È su questo argomento che vogliamo insistere su questo e nei prossimi numeri di ioProgrammo. Ad ogni applicazione la propria tecnologia, mantenere il livello di complessità più semplice possibile, trovare uno stile di programmazione facile e manutenibile. È questo l'invito che vi facciamo. È così che risolveremo grandi sfide in modo veloce ed efficace

All'inizio di ogni articolo, troverete un simbolo che indicherà la presenza di codice e/o software allegato, che saranno presenti sia sul CD (nella posizione di sempre *\soft\codice* e *\soft\tools*) sia sul Web, all'indirizzo http://cdrom.ioprogrammo.it.

ESTENDI IL TUO BLOG

Ecco come usare le API di Wordpress per creare plugin personalizzati

- ✔ **ANATOMIA DI UN ADDON** Crea uno scheletro vuoto e rendilo visibile nel pannello di amministrazione
- ✔ **HOOK E FILTRI** Intercetta gli eventi che accadono sul sito e reagisci di conseguenza
- ✔ **GESTIONE** Aggiungi una pagina nell'interfaccia privata e usala per configurare le opzioni dell'addon
- ✔ **ESEMPIO PRATICO** Ecco un plugin che si connette ad un account di Gmail e pubblica sul blog le email flaggate con una star

E?

RI-PROGRAMMA IL TUO CELLULARE

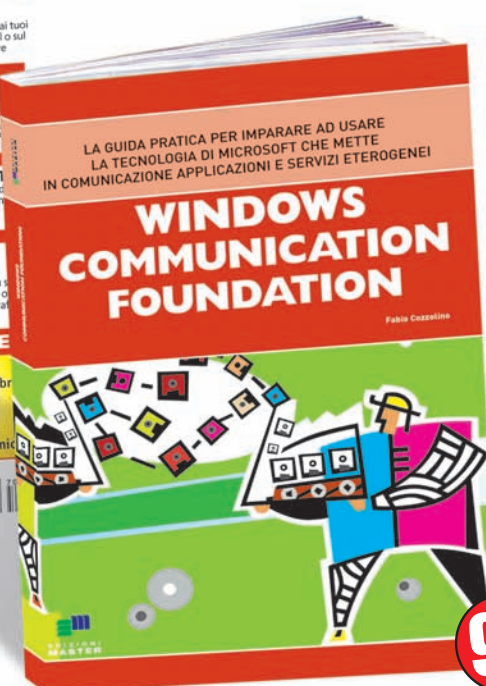
Intercetta una chiamata, un sms, o qualunque altro evento e gestiscilo come vuoi tu. Ad esempio stacca il Bluetooth se la carica della batteria e troppo bassa...

pag. 24

QUALCHE CONSIGLIO UTILE

I nostri articoli si sforzano di essere comprensibili a tutti coloro che ci seguono. Nel caso in cui abbiate difficoltà nel comprendere esattamente il senso di una spiegazione tecnica, è utile aprire il codice allegato all'articolo e seguire passo passo quanto viene spiegato tenendo d'occhio l'intero progetto.

http://forum.ioprogrammo.it

RIVISTA + LIBRO + CD-ROM in edicola

I contenuti del libro

WINDOWS COMMUNICATION FOUNDATION

In principio c'erano le applicazioni. Vivevano da sole sul PC dell'utente che le usava e non avevano bisogno di comunicare. Poi è nato il TCP/IP ed il mondo è cambiato. Il concetto di applicazione distribuita è diventato di uso comune. Il software oggi si compone di microparticelle elementari che "vivono" in contesti fisicamente distanti fra loro. Ciascuna particella comunica con le altre, fornendo un servizio all'ecosistema che nell'insieme costituirà l'applicazione. Sorge il problema di come mettere in comunicazione tutti questi componenti che spesso non rispondono ad uno standard e che utilizzano il mezzo di trasporto TCP/IP con modalità proprie. La risposta di Microsoft a questo problema si chiama "Windows Communication Foundation". Si tratta di un framework molto evoluto che consente di scrivere applicazioni e servizi che comunicano fra loro, pur lasciando al programmatore la massima libertà nella definizione degli standard di comunicazione. Questo Handbook è un riferimento affascinante per la creazione di applicazioni Interconesse.

LA GUIDA PRATICA PER IMPARARE AD USARE LA TECNOLOGIA DI MICROSOFT CHE METTE IN COMUNICAZIONE APPLICAZIONI E SERVIZI ETEROGENEI

- **Che cosa è e quando usare Windows Communication Foundation**
- **Comprendere i "contratti fra servizi eapplicazioni**
- **Gestioni delle istanze e delle applicazioni**

RIVISTA + CD-ROM
in edicola

REALBASIC 2007

Il multipiattaforma che funziona

Il sogno di molti è avere a disposizione un ambiente semplice, potente e RAD che produca eseguibili per le piattaforme Windows, Macintosh, Linux. Questo sogno oggi è realtà e si concretizza in un unico nome: Realbasic 2007. Realbasic è un prodotto straordinario. Un ambiente RAD disponibile su tutte le piattaforme che fa da cappello ad un compilatore altrettanto multipiattaforma. Il linguaggio che fa da sfondo a tutto questo è un Basic avanzato, ad oggetti ovviamente. Abbiamo provato questa versione 2007 sulle nostre macchine e ne siamo rimasti grandemente impressionati, per la velocità, la semplicità e la completezza dell'ambiente. Inoltre Real Basic è quasi completamente compatibile con il vecchio codice Visual Basic. Questo vi consente di fare diventare le vostre applicazioni VB multipiattaforma in modo estremamente semplice e veloce

Come usare l'interfaccia del CDRom

IL SOFTWARE Una accurata recensione dei contenuti

IN EVIDENZA Il top software del mese individuato dalla redazione

IL SOFTWARE Il software diviso in categorie per una comoda consultazione

HOME Torna alla pagina iniziale del CD-ROM

CONTATTACI Vuoi inviare una email alla redazione con le tue richieste

TOP SITES I siti più interessanti del mese selezionate per te

RICERCA SOFTWARE Il database di tutti i software pubblicati da ioProgrammo anche gli arretrati

IL SOFTWARE L'elenco del software contenuto nelle categorie

DIMENSIONE La dimensione del software sul CD

SALVA Clicca qui per installare o salvare il software sul tuo PC

INFO Abbonamenti informazioni e servizi utili

Il software di ioProgrammo - n. 116 - Luglio 2007

www.ioprogrammo.it
CD-ROM
ioPROGRAMMO
Anno XI - n.7 (116) - Luglio 2007
tiscali.

In evidenza
Tools
Indispensabili
Codice
Richiesta Software
Microsoft Webcast

Struts 2.0.6
Thinwire 1.2

Borland Jbuilder 2005 Foundation
L'eccezionale ambiente di Borland per lo sviluppo Java

Jbuilder è l'ambiente proposto da Borland per lo sviluppo di applicazioni Java. Come è tradizione di Borland si tratta di un ambiente incredibilmente curato nei particolari. Dotato di tutte le caratteristiche essenziali per un ottimo editor Java, la versione Foundation ha dalla sua parte il costo praticamente nullo, dovuto alla sua licenza, l'uso della foundation è infatti gratuito per usi personali. Il software è completo e senza scadenza.

Nel CD: JbuilderFoundation
Dimensione: 81,0 MB

Salva

EDIZIONI MASTER

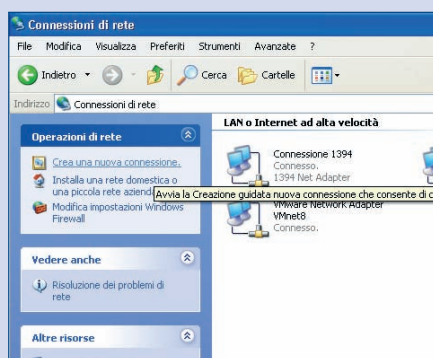
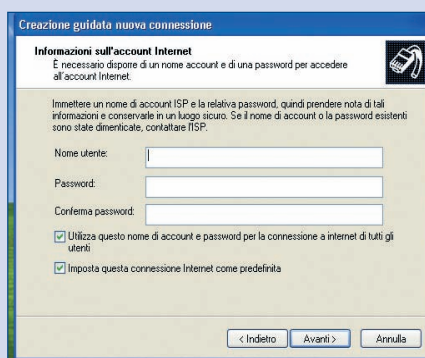
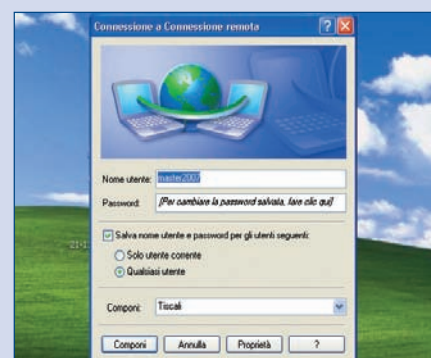
HOME | CONTATTACI | TOP SITES | RICERCA SOFTWARE | INFO | ESCI

Online con Tiscali Easy

Numero unico per tutta l'Italia e nessuna registrazione. Il modo più semplice e veloce per entrare in Internet risparmiando

Sei spesso lontano da casa e hai necessità di collegarti a Internet ovunque ti trovi? Desideri salvaguardare la tua privacy navigando in modo anonimo? Non hai voglia di perdere tempo in lunghe registrazioni per creare un nuovo account? Niente paura, Tiscali ha pensato anche a te! Con Tiscali Easy arriva un sistema tutto nuovo di collegarsi a Internet. Non sarà più necessario creare un nuovo account e fornire i propri dati personali, potrai navigare collegandoti con un unico numero di telefono (7023456789) da tutta Italia e soprattutto utilizzando i dati di accesso forniti direttamente da Tiscali (UserID e Password presenti sulla scheda), quindi non collegabili in alcun modo a te. Insomma, con Tiscali Easy, otterrai in un colpo solo riservatezza dei dati, risparmio del tempo necessario alla creazione di un abbonamento e tariffe vantaggiose. I costi di connessione sono simili a quelli di un classico abbonamento gratuito: 1,90 cent. per i primi 10 minuti e 1,72 cent. per quelli successivi. Per risparmiare ulteriormente è possibile connettersi a Internet durante le ore serali o nei giorni festivi al costo di 1,09 cent. per i primi dieci minuti e 0,98 cent. per quelli successivi. L'unico requisito richiesto per poter eseguire la connessione è un modem correttamente installato. Poiché si tratta di una normale connessione analogica è possibile utilizzare i tool di accesso remoto integrati in Windows

IN RETE CON TISCALI

Nessuna registrazione basta creare una nuova connessione e inserire i dati di accesso forniti da Tiscali

1 NUOVA CONNESSIONE Portiamoci nel pannello di controllo, e selezioniamo l'icona connessioni di rete. In alto a sinistra selezioniamo "nuova connessione" e prepariamoci a seguire il wizard che comparirà

2 DATI DI ACCESSO Seguiamo il Wizard fino a quando non ci vengono chiesti i dati per l'autenticazione. E' sufficiente inserire quelli presenti nella card allegata a questo numero di ioProgrammo: master2007/tiscali

3 ACCESSO A INTERNET Connettersi è semplicissimo, troveremo una nuova icona all'interno del pannello di controllo alla voce connessioni. Bastano due click ed il gioco è fatto sarete connessi ovunque vi troviate

News

DISPONIBILE LA RCO DI WINDOWS SERVER 2008

È stata appena rilasciata per i beta tester e gli sviluppatori la prima release candidate di Windows Server 2008. Nei progetti della casa di Redmond il nuovo sistema dovrebbe definitivamente conquistare il mercato dei server. Molta enfasi viene riservata al nuovo wizard server manager che dovrebbe aiutare nella configurazione generale del server. Particolare rilievo è stata data anche alla power shell, ovvero lo strumento a linea di comando che, sulla falsa riga, della bash dovrebbe consentire lautomazione del sistema operativo a mezzo di script. Infine l'altra news di rilevo riguarda l'introduzione del sistema "Server Core" che dovrebbe consentire un'installazione semplificata con una maggiore selezione dei componenti necessari alla sola installazione del sistema senza interfaccia grafica, per coloro che necessitano solo di un controllo remoto leggero. La RC0 di Windows Server 2008 è disponibile all'indirizzo: *http://www.microsoft.com/windowsserver2008/audsel.mspx*

IN ARRIVO HIBERNATE SEARCH 3.0

Il popolarissimo framework per la gestione dei dati si completa, tirando fuori dal cilindro un nuovo sistema per la ricerca full text. Chi volesse approfondire o scaricare il prodotto può visitare *http://www.hibernate.org/410.html*. Hibernate è nato come sistema di Object-Relational-Mapping, ovvero un framework che consente di mappare su classi ed oggetti gli elementi appartenenti ad un database relazionale. Nel tempo ha raggiunto dimensioni colossali diventando un sistema completo per la gestione dei dati. Questa nuova funzionalità di full text search che si integra completamente nel sistema principale rappresenta l'ennesimo passo avanti verso una completezza che rende sempre di più Hibernate un colosso nel suo genere

I PROGRAMMATORI HANNO BISOGNO DEL SEMANTIC WEB

Enrica Garzilli - http:/Orientalia4All.net

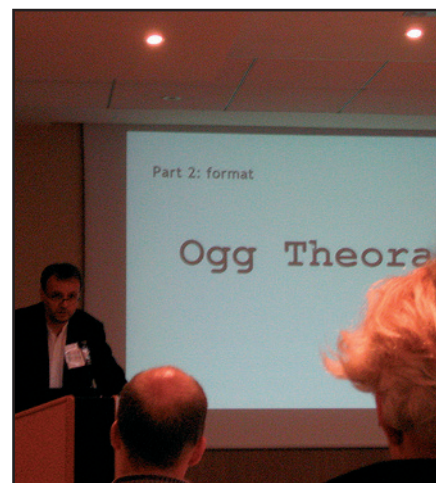
La semantica è una branca della linguistica che studia il significato delle parole, degli insiemi di parole, delle frasi, dei testi interi o, addirittura, degli insiemi di testi -- quelli, per esempio, che fanno parte dello stesso canone religioso. In questo ambito, la semantica studia il rapporto che c'è fra un parola, una frase, un testo o un insieme di testi e la realtà extralinguistica. Per esempio, la parola "ricchezza" ci fa pensare a una certa realtà se la parola è usata nel 1600 o ai giorni nostri, se è detta in una frase di contenuto filosofico o in una frase di analisi finanziaria, se viene ripetuta molte volte (in che contesto? Vicino a quali parole ricorrenti? Con che frequenza?). Quindi, la parola "ricchezza" ha molti significati, anzi, possiamo dire che nel nostro cervello ha un significato relativo a tanti altri parametri. Anche chi scrive un codice per Internet -- che compie un atto creativo come quello di scrivere un saggio ben fatto e che è simile all'insieme delle frasi di un romanzo o di un saggio, con un linguaggio ben definito che usa una sintassi sua propria -- deve tenere in conto la semantica del Web. Il Semantic Web descrive le relazioni fra le cose -- per esempio A è parte di B -- e la proprietà delle cose, per esempio la grandezza, l'età, il prezzo e così via. Ovviamente, i dati sono catturati dalle macchine e per Semantic Web si intende quello che le macchine possono fare con questi. Il Semantic Web usa principalmente il Resource Description Framework (RDF), che è un linguaggio di markup, per descrivere le informazioni e le risorse. Mettendo le informazioni nei file RDF è possibile ai web spider cercare, scoprire, collezionare, analizzare e elaborare il dato, cioè l'informazione, su Web. Se le informazioni su dei libri, per esempio, sono messe nei file RDF, delle applicazioni Web intelligenti sono in grado di collezionare questi dati da molte fonti diverse, combinarli e offrirli in modo intelligente: per esempio, potrebbero dire il prezzo di un libro, tutte le recensioni pubblicate su quel libro, paragonare prezzo e pagine di tutti i libri di un autore. Il creatore del Semantic Web è Tim Berners-Lee, l'inventore, insieme a Robert Cailliau, del World Wide Web e direttore del World Wide Web Consortium (W3C), la più grande organizzazione per gli standard su Web. Nel 1999 ha lanciato così il Semantic Web: "Ho un sogno per il Web [in cui i computer] diventeranno capaci di analizzare tutti i dati su Web – contenuti, link, e le transazioni fra persone e computer. Un Semantic Web che renda tutto questo possibile deve ancora nascere ma, quando lo farà, i meccanismi giornalieri di scambi, burocrazia e le nostre vite di giornaliere saranno gestite da macchine che parlano a macchine." Il Semantic Web, descrivendo la relazione fra le cose -- proprio come la relazione fra le parole rende possibile il loro esatto significato -- e la proprietà delle cose, di fatto analizza contenuti. Per ora, il limite del Semantic Web è che i contenuti Web sono leggibili dalla macchina, ma non necessariamente intelligibili. Un linguaggio metaforico o figurato non sarà capito. Come farà una macchina a "capire" l'espressione "Il buon senso è il sale della terra"? Il problema di RDF è che è una sintassi XML, usa elementi diversi dall'HTML (o XHTML) usato per i contenuti del web ed è un po' complicato per un'adozione di massa. In maniera meno generalizzata la stessa cosa fanno i microformati, che utilizzano lo stesso linguaggio e gli stessi elementi dell'XHTML, ma in modo strutturato. I microformati sono facilissimi da implementare per un programmatore, rappresentano un buon compromesso e sono utilizzati in molti servizi e applicazioni web. Ma perché un programmatore deve conoscere il Semantic Web? Per scambiare dati con applicazioni web, per esempio interpretando una pagina di appuntamenti o di "contact us" per estrarre dati riutilizzabili e, soprattutto, perchè ormai quasi tutte le applicazioni che hanno bisogno di interazione con le persone usano un'interfaccia web -- sia pubblica, su Internet, che privata, sulle Intranet aziendali.

GIOCA CON LA NAZIONALE DI RUBY

Enrica Garzilli

Il 26-27 ottobre si terrà a Pisa “Rails to Italy”, la prima conferenza italiana di Rails on Ruby. L’evento è organizzato da Cost e da net7 col patrocinio del Dipartimento di informatica dell’Università di Pisa, che mette anche a disposizione le aule dove si terranno le presentazioni. Ruby on Rails è un framework per il web open-source che è ottimizzato “per la felicità dei programmatori e una produttività sostenibile”, come spiega la loro homepage (*http://www.rubyonrails.org/*). I relatori invitati a parlare a “Rails to Italy” sono di tutto rispetto, come il danese David Heinemeier Hansson (in videoconferenza), programmatore ed evangelista dell’approccio Less Software, che coordina lo sviluppo di Ruby on Rails, e l’irlandese Eyal Oren, ricercatore e studente di dottorato il cui principale argomento di ricerca sono le tecniche data-centriche per la manipolazione, l’analisi e l’utilizzo di dati sul Semantic Web. Peccato solo che ancora non ci sia il programma definitivo di tutti quelli che parleranno. La particolarità di questa conferenza è che contemporaneamente si terrà anche un Coding Challenge, una gara di programmazione a tema. Gli hacker dovranno programmare "Your Favourite Thing/quello che ti piace", cioè creare un'applicazione web “piccola e interattiva”, che sia legata alla loro cosa o attività preferita. Recita l’annuncio della conferenza: “L'applicazione può iniziare chi non ne sa niente alla tua attività preferita (hai sempre voluto una simulazione interattiva della pesca del tonno?). O può essere qualcosa che è di aiuto alla tua attività (non hai sempre desiderato mappare i reperti archeologici della tua zona?). O, anche, può servire a svolgere online quello che più ti piace.” Ci sembra un po’ improbabile che un archeologo sia in grado di scrivere un programma per mappare i reperti archeologici della zona in cui vive, ma non si sa mai. Terremo d’occhio l’evento e, se troveremo degli archeologi che siano contemporaneamente dei veri hacker, anche se non vinceranno la sfida assegneramo loro un premio esclusivo: un’intervista nel nostro giornale!

THEORA BATTERÀ DIVX?

Su DivX c'è veramente poco da dire, il formato di compressione più conosciuto al mondo ha decisamente cambiato la storia dell'industria informatica e cinematografica nel bene e nel male. Sono in molti a combatterlo, le major per prime, e sono in moltissimi ad amarlo e ad usarlo. Nel corso degli anni a DivX si sono aggiunti altri formati di compressione che hanno raggiunto anche essi una notevole popolarità, ad esempio XviD, senza però mai battere il padre di tutti: il DivX. Oggi è il turno di un nuovo formato di compressione: Theora, che sta già entusiasmando gli addetti ai lavori. A dire il vero Theora non è affatto nuovo, il progetto infatti è già al suo settimo rilascio ed è sviluppato da Xiph.org. Questa nuova release ha però il pregio di essere considerata la prima beta ufficiale. Sembra uno scherzo, ma è veramente un passo da gigante se si pensa che prima di questa nuova versione sono state rilasciate ben sette release Alpha. Al di la del puro aspetto tecnologico però, sul quale gli addetti ai lavori mantengono qualche dubbio, c'è da dire che Theora persegue lo scopo di rilasciare il codice sotto forma OpenSource e quindi fornire agli sviluppatori uno strumento invidiabile per compressione di file video libera e priva di diritti di licenza. XviD difatti che viene anche esso diffuso in modo opensource rappresenta comunque un'implementazione di Mpeg-4 ed in tal senso un uso al di fuori della sfera personale potrebbe prestarsi a più di un dubbio. Theora secondo i bene informati non ha le stesse caratteristiche tecniche dei rivali, anzi per certi aspetti si rivelerebbe leggermente inferiore, tuttavia il rilascio della prima beta ufficiale rappresenta un passo importante, sia per il mercato a cui si rivolge, sia perché la disponibilità del codice consentirà a molti di fornire il proprio contributo. Di tutto ciò sicuramente non godranno le major cinematografiche, che d'altro canto combattono con il fenomeno della pirateria già da tempo. L'intero codice sorgente di theora è comunque disponibile all'indirizzo *http://www.theora.org/*.

SCRIVI UN PLUGIN PER IL TUO BLOG

ECCO COME UTILIZZARE LE API DI WORDPRESS PER AGGIUNGERVI NUOVE FUNZIONALITÀ SFRUTTANDONE LA MODULARITÀ. COME ESEMPIO PRATICO SVILUPPEREMO UN ADDON CHE SI CONNETTE AD UN ACCOUNT GMAIL E PUBBLICA LE EMAIL FLAGGATE CON UNA STAR

Molti di voi probabilmente conosceranno Wordpress: http://www.wordpress.org. Si tratta "semplicemente" del software più usato al mondo per la creazione di Blog ed è un applicazione Opensource basata su PHP.
I punti di forza di Wordpress sono molteplici:

- anche un utente poco esperto può facilmente installare un blog e gestirlo
- è molto leggero,
- i requisiti essenziali sono PHP, MySQL e un server Apache.

Dopo la prima installazione avrete a disposizione un'interfaccia di amministrazione tramite la quale scrivere i vostri post e pubblicarli, moderare i commenti, modificare le opzioni. Oltre a tutto questo avrete a disposizione un menu "Plugin".

CHE COSA È UN PLUGIN DI WORDPRESS?

Sostanzialmente è una miniapplicazione che si aggancia dinamicamente al sito principale per estenderne le funzionalità. Per installare un plugin in Wordpress è sufficiente copiare i file PHP che lo contengono nella directory *wp-contents/plugin* e attivarlo proprio dal menu "plugin" dell'interfaccia di amministrazione.
Un plugin di Wordpress può assolvere alle funzioni più disparate, l'unico limite è la fantasia di chi lo sviluppa. Ad esempio è possibile implementare una chat accessibile solo per gli utenti registrati, oppure modificare l'inserimento dei commenti affinché sfrutti Ajax, aggiungere la funzionalità di correlazione dei contenuti affinché sotto i vari post compaiono tutti quelli ad essi correlati e così via.
In questo articolo realizzeremo un plugin per Wordpress che scansiona la vostra casella postale su Gmail e pubblica sul vostro sito web le email flaggate con una star.

Per quanto riguarda la parte relativa a Gmail sfrutteremo una libreria ben conosciuta: *libgmailer*, e un lavoro opensource disponibile su internet all'indirizzo: http://ion.suavizado.com/data/files/Hacks/gallina/index.html dallo strano nome di Gallina.
Attualmente non ci interessa sapere come avviene la connessione a Gmail, il fetch dei messaggi etc. per cui ci soffermeremo veramente poco su questa parte. L'intero articolo ha il solo scopo di mostrare quali tecniche si usano per sviluppare un plugin per Wordpress, l'esempio in questione è semplicemente didattico.

STRUTTURA DI UN PLUGIN

Per prima cosa creiamo una nuova directory all'interno di *wp-contents/plugin* e chiamiamola *ioPplugin*. In questa directory inseriamo il file index.php contenente le seguenti righe:

```
<?php
/*
Plugin Name: Gmail Posting
Plugin URI: http://www.ioprogrammo.it
Description: Automatic post from google gmail -
based on a work of Jonathan Hernandez
<ion@gluch.org.mx>  - http://ion.suavizado.com/
data/files/Hacks/gallina/index.html
Version: The plugin's Version Number, e.g.: 1.0
Author: jaco
Author URI: http://www.ioprogrammo.it
Copyright 2007 Fabio Farnesi
(email : ffarnesi@edmaster.it)
This program is free software; you can redistribute it
and/or modify it under the terms of the GNU General
Public License as published by the Free Software
Foundation; either version 2 of the License, or
(at your option) any later version. This program is
distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the
implied warranty of MERCHANTABILITY or FITNESS
```

```
FOR A PARTICULAR PURPOSE.  See the GNU General
Public License for more details. You should have
received a copy of the GNU General Public License
along with this program; if not, write to the Free
Software Foundation, Inc., 51 Franklin St, Fifth Floor,
Boston, MA  02110-1301  USA
*/
?>
```

In molti guarderanno a queste righe come un semplice commento. Di fatto tutto è racchiuso fra '/*' e '*/' eppure è fondamentale inserire queste righe nel vostro file index.php.
È proprio questa struttura che rende il vostro plugin compatibile con l'architettura di Wordpress.
Salvate il file e portatevi nel menu di amministrazione e in particolare nella sezione plugin. Se tutto è andato a buon fine noterete una nuova entry nell'elenco dei plugin disponibili. Semplicemente attivate quello che avete programmato. Ovviamente il vostro plugin attualmente non fa proprio niente. Non abbiamo ancora inserito nessuna funzionalità. Abbiamo creato uno scheletro vuoto e lo abbiamo "agganciato" a Wordpress.

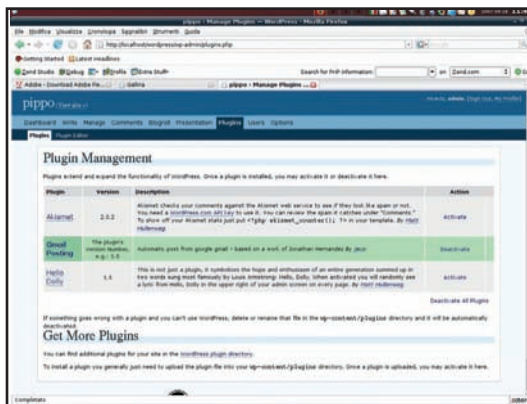



Fig. 1: Il nuovo plugin attivato nell'interfaccia dell'amministrazione

HOOK ED EVENTI

Qui arriva la parte leggermente più complicata di tutta la faccenda. Riassumiamo per punti il comportamento che vogliamo ottenere dalla nostra applicazione. Ogni volta che un utente legge una pagina del sito:

1) Viene richiamato il nostro plugin
2) Il plugin si connette ad un account di gmail
3) Vengono scansionati i messaggi e selezionati quelli flaggati con una star
4) I messaggi idonei vengono postati nel database di MySQL
5) Viene tolta la star ai messaggi già postati di modo che non ci siano duplicati

I punti **3** e **5** attualmente non ci interessano. Saranno *libgmailer* e il piccolo programmino opensource che stiamo utilizzando a svolgere questo compito. Ci interessa invece:

1) Dire al nostro sito che il plugin deve essere attivato ogni volta che un utente visita una pagina
2) Postare i messaggi nel database di MySQL

A margine di tutto ciò dovremmo fare in modo di inserire una pagina di opzioni all'interno del menu di amministrazione e che ci consenta di inserire login e password per l'accesso all'account di Gmail.
Per quanto riguarda il punto **1)**, Wordpress mette a disposizione nelle proprie API un meccanismo di Hook e Filtri. Sostanzialmente ogni volta che un evento viene scatenato, ad esempio, la visualizzazione di una pagina è possibile associarvi una funzione personalizzata. Nel nostro caso aggiungeremo semplicemente al file index.php la seguente riga:

```
add_action ( 'wp_head', $g->updatePost());
```

Questa riga significa: associa all'evento '*wp_head*' di wordpress il metodo *updatePost()* dell'oggetto *$g*. L'evento *wp_head* viene scatenato ogni volta che una pagina viene richiamata da un utente. Esistono moltissimi eventi disponibili in Wordpress, ad esempio quelli associati al post di un commento, oppure alla sua visualizzazione e così via. Nel nostro caso abbiamo usato *wp_head* per la sua semplicità, certamente non è performante, in quanto in questo modo ogni pagina vista comporta una connessione a Gmail con il conseguente delay nella visualizzazione, ma per i nostri scopi rende perfettamente l'idea.
È importante capire che ogni singola azione su un blog Wordpress è gestibile tramite un evento. Una lista completa degli eventi disponibili nelle

NOTA

LA LIBRERIA LIBGMAILER

Si tratta di un progetto PHP che implenta una serie di classi che facilitano l'accesso ad un account Gmail. Il sito originale del progetto è http://gmail-lite.sourceforge.net/ La libreria in questione è stata utilizzata nel progetto Gallina per la connessione e il fetch delle informazioni

IL PROGETTO GALLINA

Per quanto il nome possa sembrare ironico, curioso e persino divertente per molti, questo progetto è piuttosto interessante. Si tratta di un'applicazione PHP che si connette ad un account gmail e salva in file XML le email flaggate con una star. Il file XML in questione viene poi rielaborato tramite XSL e produce un vero e proprio Blog. Il sito originale del progetto da cui potete anche scaricare i file che sono stati poi modificati per questo articolo è http://ion.gluch.org.mx/files/Hacks/gallina/

API di WP è pubblicata su questo sito: http://codex.wordpress.org/Plugin_API#Hooks.2C_Actions_and_Filters
Sempre a proposito del nostro esempio appare ovvio che non abbiamo ancora instanziato l'oggetto *$g* e ovviamente non abbiamo creato nessun metodo *updatePost()*. Non ci siamo connessi a Gmail etc, in questo paragrafo abbiamo solo voluto spiegare come funziona il meccanismo degli hook.

MAGGIORI DETTAGLI

Tanto per rendere quasi completamente funzionante la nostra applicazione aggiungiamo qualche altra riga di codice all'index.php. Dopo le righe necessarie per rendere visibile il plugin, completiamo il nostro index.php come segue:

```
require_once "include/Gallina.php";
$g = new Gallina (get_option('g_email'),
                  get_option('g_passwd'), '0');
$g->Connect ();
add_action ( 'wp_head', $g->updatePost());
```

Con la prima riga richiamiamo la piccola libreria opensource che stiamo utilizzando per la connessione a Gmail. Ovviamente dovremo aver copiato i file necessari nella sottodirectory "*include*" del plugin che stiamo sviluppando.
Con la seconda riga instanziamo l'oggetto *$g* che ci servirà per la connessione ed il fetch dei messaggi. Con la terza riga effettuiamo la connessione, infine inseriamo il nostro Hook che allo scatenarsi dell'evento *wp_head* richiama il metodo *updatePost()*. Il metodo *updatePost()* è quello che fisicamente si occuperà di recuperare le email flaggate con una star da Gmail e postarle sul blog. In tutto questo nella seconda riga non sappiamo ancora come vengano recuperate password e login per la connessione a Gmail, ma ce ne occuperemo più tardi.

IL METODO UPDATEPOST

Non ci soffermeremo a lungo su questo metodo, perché non riguarda la didattica sullo sviluppo di un plugin per Wordpress ma è semplicemente un hack della libreria che stiamo usando. Implementeremo questo metodo nel file gallina.php come segue:

```
function updatePost ()
{
        $info = $this->GetInfo ();
        $entries = $this->GetRecentEntries ();
        //// my first object :-)
        $wm_myobject = new wm_mypost();
        // fill object
        // feed object to wp_insert_post
        foreach ($entries as $e) {
                $wm_myobject->post_title =
                        $this->PrepareText ($e->Title);
$wm_myobject->post_content =$e->Body;
$wm_myobject->post_status = 'publish';
$wm_myobject->post_author = 1;
wp_insert_post($wm_myobject);
                }
}
```

Di tutto questo metodo l'unica cosa che ci riguarda da vicino è relativa al posting dei messaggi. Viene instanziato un oggetto *wm_myobject* di classe *wm_mypost*. Questo oggetto viene riempito con le informazioni contenute nell'array *$entries*, ovvero *soggetto* e *titolo* di una mail e infine viene passato alla funzione *wp_insert_post* che si occupa fisicamente di inserire i dati in questione nel db di MySQL e pubblicarli sul blog. Ovviamente non ci interessa il come il metodo *GetRecentEntries()* riempia l'array *$entries* con le informazioni che ci servono, piuttosto ci interessa sapere come è composta la classe *wm_mypost*. La definiremo poco prima sempre in gallina.php come segue

```
class wm_mypost {
        var $post_content;
        var $post_title;
        var $post_status;
}
```

In definitiva la funzione *wp_insert_post()* che è resa disponibile direttamente dalle API di Wordpress accetta come parametro un oggetto al cui interno siano definite le variabili che contengono i dati da postare. È esattamente quello che abbiamo fatto noi.

AGGIUNGERE UN MENU

Non ci rimane che stabilire come passare i parametri di connessione necessari per connettersi all'account di gmail

```
$g = new Gallina (get_option('g_email'),
                  get_option('g_passwd'), '0');
$g->Connect ();
```

get_option è appunto la funzione di wordpress che recupera dal database MySQL il contenuto di eventuali opzioni in esso definite.

Per cui dovremo semplicemente salvare nel database le opzioni *g_email* e *g_passwd* che conteranno al loro interno rispettivamente il nome di login e la password per l'account di Gmail.
Per fare questo avremo bisogno di una nuova pagina da visualizzare all'interno del menu di amministrazione di Wordpress e che visualizzi una form che ci consenta di immettere le informazioni desiderate. Dovremo poi, ovviamente, implementare i meccanismi che consentano di persistere queste informazioni nel database di Wordpress.
Mettiamoci all'opera. Prima di tutto aggiungiamo un hook che faccia in modo che ogni volta che viene richiamata la pagina di amministrazione di Wordpress vi aggiunga una voce. Possiamo farlo semplicemente come segue:

```
add_action('admin_menu', 'mt_add_pages');
```

Questa riga specifica che ogni volta che viene richiamata la pagina di amministrazione di WP deve essere eseguita la funzione *mt_add_pages* andiamo perciò a definire il comportamento di questa funzione
0

```
function mt_add_pages() {
    add_options_page('Gallina Options', 'Gallina
      Options', 8, 'gallinaoptions', 'mt_options_page');
}
```

Grazie ad *add_options_page* che è una funzione resa disponibile dalle API di Wordpress la *mt_add_pages* aggiunge una voce al menu di amministrazione ed in particolare nel sottomenu Options. Questa voce sarà visibile come 'Gallina Options', se cliccata avrà come titolo di pagina 'Gallina Options', sarà accessibile dagli utenti con almeno un livello 8, avrà come identificativo unico 'gallinaoptions' e richiamerà la funzione *mt_options_page*.
Ovviamente la *mt_options_page* dovrà contenere al suo interno i meccanismi per la visualizzazione della form e il posting delle informazioni "g_email" e "g_password" all'interno del datase.

LA PAGINA DELLE OPZIONI

Questa è la parte probabilmente più complicata di tutta la baracca. Diamo un'occhiata alla funzione *mt_options_page*

```
function mt_options_page() {

  // variables for the field and option names
    $opt_name =
                    array(0=>'g_email',1=>'g_passwd');
    $hidden_field_name = 'mt_submit_hidden';
    $data_field_name =
                    array(0=>'g_email',1=>'g_passwd');

    // Read in existing option value from database
    $opt_val['g_email'] =
                    get_option( $opt_name[0]);
    $opt_val['g_passwd'] =
                    get_option( $opt_name[1]);
    // See if the user has posted us some information
    // If they did, this hidden field will be set to 'Y'
    if( $_POST[ $hidden_field_name ] == 'Y' ) {
      // Read their posted value
      $opt_val['g_email'] = $_POST[
                    $data_field_name[0]];
      $opt_val['g_passwd'] = $_POST[
                    $data_field_name[1]];

      // Save the posted value in the database
      update_option( $opt_name[0],
                    $opt_val['g_email'] );
          update_option( $opt_name[1],
                    $opt_val['g_passwd'] );

      // Put an options updated message on the
                    screen

      echo '<div id="message" class="updated
                    fade"><p><strong>option
                    saved</strong></p></div>';
    }

    // Now display the options editing screen

    echo '<div class="wrap">';

    // header

    echo "<h2>" . __( 'Gallina Plugin Options',
                    'mt_trans_domain' ) . "</h2>";

    // options form
echo '<form name="form1" method="post"
                    action="'.str_replace( '%7E', '~',
                    $_SERVER['REQUEST_URI']).'">';
echo '<input type="hidden"
                    name="'.$hidden_field_name.'" value="Y">';
echo '<p>'._e("Gmail account:",
                    'mt_trans_domain');
echo '<input type="text"
                    name="'.$data_field_name[0].'"
                    value="'.$opt_val['g_email'].'"size="20">';
echo '<br>';
echo '<p>'._e("password:", 'mt_trans_domain');
echo '<input type="password"
                    name="'.$data_field_name[1].'"
                    value="'.$opt_val['g_passwd'].'"size="20">';
```

```
echo '</p><hr />';

echo '<p class="submit">';
echo '<input type="submit" name="Submit"
                    value="Update Options"/>';
echo '</p>';
echo '</form>';
echo '</div>';

 }
```

Iniziamo dalle cose semplici: le linee che seguono // header visualizzano sullo schermo la form con le opzioni da immettere. Ci sono alcune note da fare rispetto a questa form. In particolare

```
echo '<input type="text"
                    name="'.$data_field_name[0].'"
            value="'.$opt_val['g_email'].'"size="20">';
echo '<br>';
echo '<p>'._e("password:", 'mt_trans_domain');

echo '<input type="password"
                    name="'.$data_field_name[1].'"
         value="'.$opt_val['g_passwd'].'"size="20">';
```

Come potete vedere voi stessi il nome del campo ed il suo valore vengono riempiti dinamicamente prelevandoli rispettivamente dall'array *data_field_name* e *opt_val*. I due array in questione vengono riempiti dalle seguenti righe

```
  $opt_name = array(0=>'g_email',1=>'g_passwd');
  $data_field_name =
               array(0=>'g_email',1=>'g_passwd');
```

Immediatamente dopo, la funzione in questione controlla se sono state già salvati dei valori per le opzioni g_email e g_passwd nel database di MySQL e se esistono valorizza l'array $opt_val

```
   $opt_val['g_email'] = get_option( $opt_name[0]);
     $opt_val['g_passwd'] = get_option(
                              $opt_name[1]);
```

Poi si controlla se la pagina è stata visualizzata in seguito a un post o se è la prima volta che viene visualizzata. Nel primo caso si mostra un messaggio di update eseguito e fisicamente si salvano i valori nel database

```
if( $_POST[ $hidden_field_name ] == 'Y' ) {
        // Read their posted value
        $opt_val['g_email'] = $_POST[
                              $data_field_name[0]];
        $opt_val['g_passwd'] = $_POST[
                              $data_field_name[1]];

        // Save the posted value in the database
        update_option( $opt_name[0],
                              $opt_val['g_email'] );
              update_option( $opt_name[1],
                              $opt_val['g_passwd'] );

        // Put an options updated message on the
                                          screen
        echo '<div id="message" class="updated
                   fade"><p><strong>option
                 saved</strong></p></div>';
    }
```

Infine viene visualizzata la form e in seguito ad un eventuale post si ricomincia il ciclo. E con questo il nostro esempio può ritenersi concluso e funzionante.

DIFFERENZA FRA FILTRI E HOOK

Nella documentazione ufficiale di Wordpress troverete che vengono espressi due tipi di eventi a cui associare le funzioni personalizzate. Si parla di Hook quando si intercettano eventi legati a wordpress stesso ad esempio la visualizzazione di una pagina e di filtri quando si vogliono applicare delle modifiche al testo al database prima che l'output venga inviato all'utente. Ad esempio se si volesse fare in modo di marcare in rosso tutte le parole inglesi bisognerebbe programmare un filtro che esegue un match con un dizionario ed in seguito ad un riscontro aggiunge i necessari tag font alle parole da evidenziare.

CONCLUSIONI

Wordpress è una delle applicazioni PHP che maggiormente nel corso degli anni hanno caratterizzato lo sviluppo della “Blogosfera”. Proprio la sua estendibilità a mezzo di plugin ne ha fatto un software così diffuso. Tra l’altro esiste un mercato molto interessante di Plugin. Ne potete trovare un esempio sul sito cdex.wordpress.org. Ne esistono praticamente di tutti i tipi e coinvolgno funzioni legate ad ogni esigenza di gestione del posting di contenuti. Curiosando su Internetr troverete inoltre molti esempi di siti commerciali sviluppati estendendo Wordpress a mezzo di plugin ed integtrando le funzionalità desiderate all’interno dei Template. In questo articoli abbiamo tralasciato di trattare l’argomento template, lo accenniamo solo qui in coda all’articolo dicendo che si tratta di normali file php che contengono un misto di grafica e programmazione. Non mancheremo di trattare l’argomento in u no dei prossimi numeri. Provate voi stessi a sviluppare un plugin e vi accorgerete di come questo meccanismo sia estremamente flessibile.

Fabio Farnesi

RI-PROGRAMMA IL TUO CELLULARE

INTERCETTA GLI EVENTI LEGATI AL TELEFONO, AD ESEMPIO UNA CHIAMATA O L'ARRIVO DI UN SMS E FAI IN MODO CHE LA TUA APPLICAZIONE REAGISCA IN MODO PERSONALIZZATO. AD ESEMPIO DISATTIVA LA VIBRAZIONE SE IL LIVELLO DELLA BATTERIA È BASSO....

L'uso di dispositivi mobili (palmari e Smartphone) come piattaforma di sviluppo per le applicazioni, ha da sempre interessato le aziende e gli sviluppatori.
Vuoi per la comodità di avere tutta una serie di informazioni a portata di mano (listini, clienti etc.), vuoi per il vantaggio di poter acquisire ordini direttamente presso il cliente, vuoi per l'estrema portabilità di questi device, l'eventualità di sviluppare applicazioni che si integrassero con i sistemi pre-esistenti è spesso stata presa in considerazione.
Quello che ha in un certo senso frenato la loro diffusione, sono stati due fattori non sottovalutabili: il costo dei device ed il costo di sviluppo su questa piattaforma.
Per il primo fattore, come si sa, l'evoluzione tecnologica porta alla realizzazione di dispositivi sempre più potenti ed a costi sempre più bassi. Se guardiamo un device di un paio di anni fa (sotto l'aspetto prestazionale ed economico), ci rendiamo immediatamente conto dell'impossibilità di paragonarlo, sotto entrambi i punti di vista, ad uno dei device oggi in commercio.
Per il secondo fattore, come per il primo, l'evoluzione tecnologica ha portato alla realizzazione di linguaggi, piattaforme e tool di sviluppo che semplificano notevolmente lo sviluppo di applicazioni anche complesse su questi dispositivi.
In questo articolo, tratteremo una serie di API introdotte con Windows Mobile 5.0 (ed ampliate su Windows Mobile 6), che semplificano di molto il lavoro degli sviluppatori, permettendo di creare software più versatili rispetto al passato, e di introdurre funzionalità davvero interessanti nelle nostre applicazioni.
Parliamo di State and Notification Broker.
Overview
State and Notification Broker API è, senza ombra di dubbio, una delle rivoluzioni che riguardano Windows Mobile 5.0 (ed il nuovo Windows Mobile 6). L'introduzione di queste API, ha sostanzialmente modificato il modo in cui, dal punto di vista degli sviluppatori, ci si può approcciare alla creazione di applicazioni sulla piattaforma Windows Mobile. L'enorme pregio di questi elementi, è quello di aver trasformato la piattaforma mobile da un sistema in cui "far girare" le applicazioni, ad un sistema con cui le nostre applicazioni possono integrarsi in modo semplice e, in alcuni casi, possono anche interagire tra loro o con il sistema operativo.
Prima della loro introduzione, le applicazioni sviluppate per dispositivi mobili erano di fatto isolate e sostanzialmente prive di interazione con il sistema operativo e con le applicazioni su esso presenti. Introdurre questa tipologia di interazione era un'operazione non sempre semplice, al punto da far desistere lo sviluppatore dall'includere determinate funzionalità nel proprio software. Funzionalità che spesso avrebbero potuto dare un valore aggiunto all'applicazione e, in alcuni casi, prevenire anche errori.
Un esempio concreto di quanto appena affermato, potrebbe essere l'elaborazione di grosse quantità di dati direttamente sul dispositivo. Lavorando su un dispositivo mobile, va considerato che la batteria di cui è equipaggiato possa scaricarsi o, peggio ancora, potrebbe essere già scarica al momento in cui l'operatore decide di avviare l'operazione lunga, causando magari inconsistenza dei dati se l'operazione si interrompe.
Uno dei valori aggiunti che potremmo dare alla nostra applicazione, potrebbe essere quello di impedire l'avvio della suddetta operazione in determinate condizioni o, ancora meglio, disabilitare le opportune voci di menù in automatico quando la batteria scende al di sotto di una soglia da noi definita critica per l'operazione in oggetto.
In passato, avremmo potuto imboccare tre strade possibili per risolvere la questione:

1. abbandonare la realizzazione di tale funzionalità: scelta preferita dai più, vista la difficoltà implementativa;
2. lavorare via Interop (Platform Invokation –

P/Invoke) andando a richiamare direttamente alcune API del sistema operativo adatte allo scopo. Una cosa certamente possibile ma non semplicissima e che comporta l'aumento della complessità dell'applicazione.

3. affidarsi a componenti di terze parti, spesso a pagamento. Anche qui, introduzione di altra complessità nell'applicazione.

Altro esempio concreto è l'intercettazione dell'orientamento dello schermo del dispositivo (portrait o landscape), funzionalità già presente in Windows Mobile 2003 SE. Volendo recuperare questa informazione via codice, avremmo dovuto richiamare *GetSystemMetrics* o controllare la proprietà *Screen.PrimaryScreen.Bounds*.
Stesso discorso qualora avessimo voluto recuperare la quantità di carica della batteria. Avremmo dovuto utilizzare *GetSystemPowerStatusEx*.
Tutti elementi di API diverse, esposte in modo diverso e non sempre di facile accesso.
Vediamo ora cosa cambia con l'introduzione di State and Notification Broker.

STATE AND NOTIFICATION BROKER

Il grande pregio delle API raccolte sotto il nome "State and Notification Broker", è qullo di fornire, in modo unificato, sia l'accesso ad una serie di informazioni di stato messe a disposizione del sistema operativo e dalle applicazioni in esso installate, sia eventualmente di notificare agli eventuali subscribers il cambiamento di valore di questi stati.
Questo sistema di accesso unificato alle funzionalità ed alle notifiche, nonché di memorizzazione delle informazioni in un'area comune, ci da la possibilità di inserire, nelle nostre applicazioni, funzionalità avanzate e, soprattutto, di farlo in modo semplice e veloce. Funzionalità che in parte erano già disponibili anche con le versioni passate di Windows Mobile, ma di sicuro, non erano di così facile accesso.
Ad oggi, State and Notification, espone in un repository comune collocato all'interno del registro di Windows Mobile, più di 100 valori relativi ad una serie di stati di Windows Mobile e delle applicazioni in esso installate.
Alcuni esempi di valori esposti sono:

- per gli stati di sistema
- **ActiveSync**: se stiamo eseguendo una sincronizzazione
- **Battery**: informazioni sullo stato delle batterie presenti sul device
- **Bluetooth**: numero delle connessioni bluetooth attive
- **Network**: lista delle connessioni alla rete attive
- **Camera**: indica la presenza di una fotocamera sul device
- **Handsfree**: indica se stiamo utilizzando un kit vivavoce
- etc.
- per gli stati relativi all'utente:
- **Appointments**: l'appuntamento corrente, quello successivo, quello precedente etc.
- **Messages**: informazioni sulla presenza di SMS, MMS, E-mail, messaggi non letti etc.
- **Media Player**: informazioni su Media Player come la traccia corrente etc.
- **Tasks**: informazioni sui tasks (le attività)
- etc.

Per la precisione, tali valori sono conservati sotto le chiavi:

- ***HKEY_LOCAL_MACHINE\System\State*** per gli stati di sistema
- ***HKEY_CURRENT_USER\System\State*** per gli stati relative all'utente corrente

Possiamo esplorare tutti gli stati a disposizione (ed i relativi valori), utilizzando un tool come ***Windows CE Remote Registry Editor*** (presente in C:\Program Files\CE Remote Tools\5.01\bin\ccregedt.exe). Con esso, possiamo esplorare il registro di sistema sia dei device collegati al nostro PC, sia, come nel caso dell'esempio, degli emulatori su cui stiamo testando l'applicazione (Figura 1).

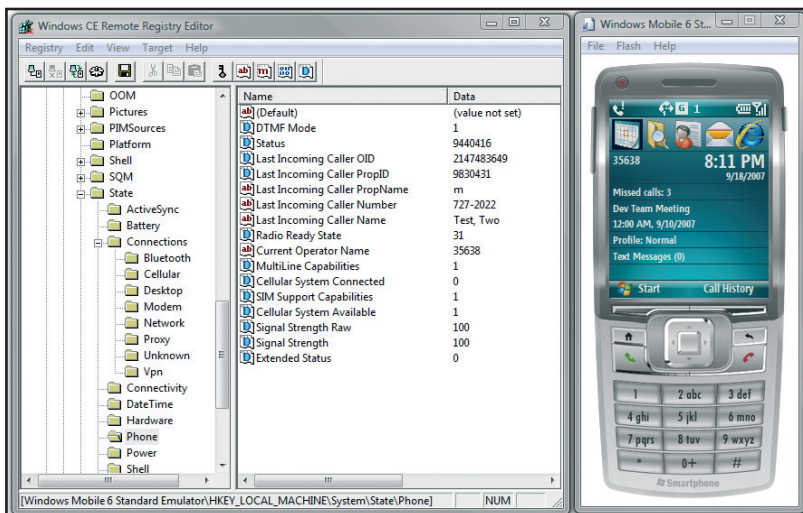



Fig. 1: La visualizzazione del registro con Remote Registry Editor

LETTURA DEGLI STATI

Dopo aver visto per grandi linee come è strutturato ed a cosa serve lo State and Notification Broker, concentriamoci su come, usando codice

LINKS UTILI
All'indirizzo http://msdn2.microsoft.com/en-us/library/bb154506.aspx **è disponibile una utile tabella in cui sono elencati tutti gli stati esposti, la relativa descrizione ed il loro tipo .NET**

.NET (in questo caso C#, ma le stesse operazioni possono essere svolte con Visual Basic .NET), è possibile accedere alle informazioni memorizzate negli stati ed usarle nelle nostre applicazioni. Operazione, tra l'altro, decisamente molto semplice.
L'uso di uno dei linguaggi supportati dal .NET Compact Framework infatti, nasconde molta della logica interna necessaria all'accesso dei valori contenuti nelle chiavi di registro. Il risultato è quello di poter accedere ai vari stati utilizzando poche righe di codice, rendendo la nostra applicazione molto leggibile e sicuramente più manutenibile.
Usando il .NET Compact Framework 2.0, l'accesso ai valore del registro avviene attraverso una serie di proprietà statiche dell'oggetto **SystemState**, contenuto all'interno del namespace **Microsoft.WindowsMobile.Status**. In Figura 2 è visibile una porzione degli stati direttamente accessibili attraverso l'intellisense di Microsoft Visual Studio 2005.

sono tipizzate, il che ci permetterà di usarle in modo molto flessibile all'interno della nostra applicazione, nonché di accedere ad ulteriori informazioni relative al tipo specifico. Per l'aggiunta del riferimento, sarà sufficiente fare click con il tasto destro del mouse nella cartella Reference (Riferimenti) del progetto in Microsoft Visual Studio .NET 2005, selezionare la voce Add Reference (Aggiungi Riferimento) e selezionare la libreria indicata.
A questo punto, siamo pronti per iniziare la lettura degli stati. Lo facciamo creando una semplice applicazione in cui mostreremo il valore della carica della batteria del dispositivo.
Per farlo, creiamo progetto in Microsoft Visual Studio .NET 2005, selezionando come tipologia di progetto Smart Device, Windows Mobile 6 Standard e come template di progetto Device Application (Vedi Figura 4). Assegnamogli un nome ed una cartella in cui salvare i file e premiamo il pulsante Ok.

Fig. 2: L'intellisense di Microsoft Visual Studio .NET 2005 sugli stati

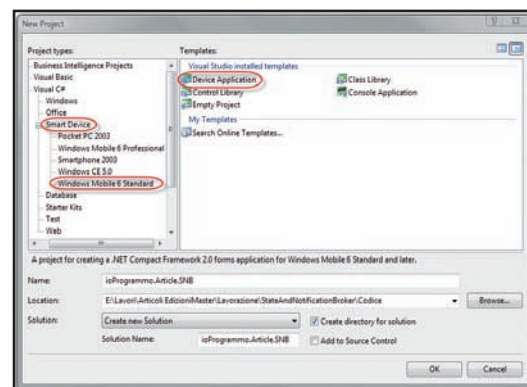



Fig. 4: Creazione del nuovo progetto in Visual Studio 2005

Prima di accedere alle informazioni sugli stati, andrà però aggiunto un nuovo riferimento al progetto. Per la precisione, andrà aggiunto **Microsoft.WindowsMobile.Status** (Figura. 3) e

Fig. 3: Aggiunta del riferimento a Microsoft.Windows Mobile.Status

successivamente leggere le proprietà **SystemState** che ci interessano. Tali proprietà

NOTA

DOVE TROVO L'SDK?

Il Microsoft Windows Mobile 6.0 SDK Refresh è scaricabile al seguente indirizzo web: http://www.microsoft.com/downloads/details.aspx?familyid=06111a3a-a651-4745-88ef-3d48091a390b&displaylang=en ed include gli emulatori ed i template di Microsoft Visual Studio .NET 2005.

Fatto questo, Visual Studio preparerà per noi la base del progetto aggiungendo, tra le altre cose, il primo form a cui, per gli scopi dell'esempio, dovranno essere aggiunte due label ed una nuova voce di menù associata ad un softKey del dispositivo. All'evento click della nostra voce di menù, aggiungiamo il seguente codice:

```
using System;
using System.Windows.Forms;
using Microsoft.WindowsMobile.Status;

namespace ioProgrammo.Article.SNB {
    public partial class Form1 : Form {
        /// <summary>
        /// Livello di carica della batteria
        /// </summary>
        private BatteryLevel _batteryLevel;

        public Form1() {
```

```
            InitializeComponent();
        }

        /// <summary>
        /// Azione associata al click sul menù Leggi
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void menuItem2_Click(object sender,
                                     EventArgs e) {
            _batteryLevel =
                  SystemState.PowerBatteryStrength;
            lblBatteryLevel.Text =
                        _batteryLevel.ToString();
        }

        /// <summary>
        /// Chiusura dell'applicazione
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void menuItem1_Click(object sender,
                                     EventArgs e) {
            Application.Exit();
        }
    }
}
```

Come è facile intuire dal codice, cliccando sul menù “Leggi”, nella label lblBatteryLevel ci troveremo una indicazione relativa allo stato della batteria principale del dispositivo. Avviando l’applicazione sull’emulatore, otterremo infatti il risultato di Figura 5.

Fig. 5: L’applicazione di esempio in esecuzione

Lo stesso procedimento si applica per tutti gli stati esposti dal sistema (e, ricordo, sono più di 100. Fare riferimento al box laterale per l’elenco completo).

NON SOLO LETTURE

Come è facile intuire dal nome (e come abbiamo anticipato in precedenza), queste nuove API non si limitano solo a fornire dei valori da poter leggere dalle nostre applicazioni (come nell’esempio precedente), ma forniscono un evoluto sistema di notifica. Il concetto su cui si basa questo sistema è molto semplice: attraverso queste API, sono esposti una serie di eventi (come ad esempio il cambio di livello della batteria). Una nostra applicazione, può “registrarsi” a questi eventi e “reagire” al cambiamento. Per poter ricevere le notifiche di cambio di stato, dobbiamo sostanzialmente eseguire tre step fondamentali:

1. **Registrarsi alle notifica di cambiamento di uno stato**: è la prima cosa da fare. Se vogliamo essere notificati circa il cambio di uno stato, come prima cosa dobbiamo “informare” il sistema operativo che vogliamo ricevere quella particolare notifica.
2. **Gestire l’evento di notifica**: una volta registrati ad un evento, nel nostro codice dobbiamo gestire la notifica relativa all’evento registrato in modo da poter compiere delle operazioni specifiche.
3. **Leggere il nuovo valore**: l’arrivo di una notifica indica che un particolare valore è stato modificato. A questo punto possiamo leggere il nuovo valore.

Riferendoci all’esempio fatto in precedenza circa lo stato di carica della batteria, vediamo come poter implementare in pochi e semplici passi un sistema che disattiva una voce di menù in caso il livello di carica della batteria scenda al di sotto di una determinata soglia. Seguiamo quindi i tre punti precedentemente elencati. Per prima cosa, dobbiamo registrare la nostra applicazione all’evento che vogliamo gestire, passando nel costruttore di SystemState, l’enumerazione relativa allo stato di nostro interesse. Subito dopo, nell’inizializzazione del form, ci registriamo all’evento Changed:

```
/// <summary>
/// Creazione dell'istanza di SystemState relativa
                          all'evento da monitorare
/// </summary>
private SystemState _batteryState = new
  SystemState( SystemProperty.PowerBatteryStrength
                                                 );
```

NOTA

FUNZIONALITÀ DELL’EMULATORE

Con i nuovi emulatori, è possibile regolare il livello di carica della batteria per effettuare i nostri test. Per farlo, sarà sufficiente cliccare sul menù “File” dell’emulatore e scegliere la voce “Configure”. Dalla maschera che di aprirà, cliccare sul tab “Peripherals”, spuntare la voce “Battery” ed impostare un valore numerico compreso tra 0 e 100.

```
/// <summary>
/// Livello ella batteria
/// </summary>
private BatteryLevel _batteryLevel =
                    SystemState.PowerBatteryStrength;

public Form2() {
   InitializeComponent();
   //Registrazione dell'evento Changed dello stato
                                           monitorato
   _batteryState.Changed += new
       ChangeEventHandler(_batteryState_Changed);
}
```

A questo punto, non ci resta che gestire l'evento in questione:

```
/// <summary>
/// Gestione dell'evento Changed
/// </summary>
/// <param name="sender"></param>
/// <param name="args"></param>
void _batteryState_Changed(object sender,
                             ChangeEventArgs args) {
   //Lettura del nuovo stato
   _batteryLevel = (BatteryLevel)args.NewValue;
   //Imposto il testo nella label con il nuovo valore
   lblBatteryLevel.Text = _batteryLevel.ToString();
   //in base al livello della batteria, abilito o
                                      disabilito il menù
   if (_batteryLevel == BatteryLevel.VeryLow) {
      menuItem2.Enabled = false;
   } else if (_batteryLevel == BatteryLevel.High ||
           _batteryLevel == BatteryLevel.VeryHigh) {
      menuItem2.Enabled = true;
   }
}
```

Fig. 6: L'applicazione di esempio in esecuzione

Il nuovo valore dello stato monitorato, ritorna attraverso gli **EventArgs** (ChangeEventArgs) dell'evento **Changed** sotto forma di Object. Prima di usare tale valore, dovremmo quindi castarlo al tipo specifico che stiamo gestendo.
Avviando l'applicazione sull'emulatore, e variando il livello della batteria (vedi box laterale), vedremo come il menù "Elabora" verrà abilitato o disabilitato in funzione del valore che andremo ad impostare. Il risultato, seppur statico, è quello visibile in Figura 6.

STATE AND NOTIFICATION BROKER: USO CONCRETO NELLE APPLICAZIONI

Fino ad ora, abbiamo visto alcuni dei concetti di base relativi a queste API. Gli esempi fin'ora mostrati, hanno avuto lo scopo di introdurre le potenzialità di questo meccanismo e di comprenderne per grandi linee il funzionamento. Per motivi di spazio, non possiamo addentrarci troppo nell'architettura e nelle funzionalità più avanzate, ma possiamo, sempre attraverso una serie di esempi pratici, toccare con mano alcune di queste funzionalità ed apprezzarne l'utilità pratica in applicazioni più complesse.
Se prendiamo la tabella con tutte gli stati esposti dal sistema (vedi box laterale), possiamo notare che alcuni degli stati riguardano la parte telefonica del device (Phonexxxxx). E' quindi facile intuire che, con estrema semplicità, possiamo includere nella nostra applicazione, funzionalità per la gestione delle chiamate.
Se volessimo ad esempio gestire le chiamate perse (Missed Call), magari per registrarle in una nostra applicazione a fini statistici, sarà sufficiente "lavorare" sullo stato PhoneMissedCall:

```
/// <summary>
/// Chiamante
/// </summary>
private SystemState _incomingCaller;

private void Form3_Load(object sender, EventArgs e)
                                                  {
   //Istanza di SystemState specializzata sullo stato
                                         da monitorare
   _incomingCaller = new
   SystemState(SystemProperty.PhoneIncomingCaller
                                              Name);
   //Evento Change
   _incomingCaller.Changed += new
     ChangeEventHandler(_incomingCaller_Changed);
```

NOTA

ESEMPI ONLINE

On line è disponibile una applicazione completa (e di cui è fornito il codice sorgente), interamente basata su State and Notification Broker. L'applicazione è liberamente scaricabile on line da qui: http://www.codeplex.com/MobilePhoneAssistant.

```
}

/// <summary>
/// Gestione dell'evento Change
/// </summary>
/// <param name="sender"></param>
/// <param name="args"></param>
void _incomingCaller_Changed(object sender,
                               ChangeEventArgs args) {
    label1.Text = args.NewValue.ToString();
}
```

All'interno del gestore dell'evento, possiamo ovviamente inserire del codice più complesso ai fini, ad esempio, di memorizzare la chiamata persa all'interno di un Data Base.
Questo semplice esempio, serve per introdurre due concetti molto interessanti di State and Notification Broker: la specializzazione delle notifiche e le notifiche persistenti.
Ricollegandoci all'esempio precedente, come è facile intuire dal codice, la nostra applicazione "reagirà" a tutte le chiamate in arrivo provenienti da un numero registrato nella rubrica del dispositivo (stiamo usando PhoneIncoming CallerName). Possiamo però specializzare le notifiche (di qualsiasi tipo) applicando loro un filtro. Tale specializzazione ha come scopo quello di restringere il range di notifiche inviate all'applicazione solo a determinati valori da noi imposti. L'applicazione del filtro avviene attraverso l'impostazione di due proprietà specifiche dell'oggetto **SystemState: ComparisonType e ComparisonValue.**
La propietà **ComparisonType** rappresenta il tipo di filtro che vogliamo applicare alla notifica mentre l'impostazione di questa proprietà avviene attraverso l'enumerazione **StatusComparisonType** i cui valori sono elencati in **tabella 1.**
Vediamo ad esempio come filtrare l'applicazione precedente solo per le chiamate provenienti dal contatto "Test, Two". Per farlo, sarà sufficiente aggiungere, al gestore dell'evento load del form visto nel precedente esempio, i criteri di filtro:

```
private void Form3_Load(object sender, EventArgs e)
                                                  {
    //Istanza di SystemState specializzata sullo stato
                                          da monitorare
    _incomingCaller = new
     SystemState(SystemProperty.PhoneIncomingCaller
                                                Name);
    //Imposto il tipo di comparazione
    _incomingCaller.ComparisonType =
                              StatusComparisonType.Equal;
    //Imposto il valore da filtrare
    _incomingCaller.ComparisonValue = "Test, Two";
```

```
    //Evento Change
    _incomingCaller.Changed += new
      ChangeEventHandler(_incomingCaller_Changed);
}
```

Da questo momento, la label dell'esempio verrà aggiornata solo se il chiamante è quello indicato nel filtro.

Enumerazione	Descrizione
Equal	Il valore presente nel registro al momento della notifica è uguale a quello impostato nella proprietà ComparisonValue
NotEqual	Il valore presente nel registro al momento della notifica è diverso da quello impostato nella proprietà ComparisonValue
Greater	Il valore presente nel registro al momento della notifica è maggiore da quello impostato nella proprietà ComparisonValue
GreaterOrEqual	Il valore presente nel registro al momento della notifica è maggiore o uguale a quello impostato nella proprietà ComparisonValue
Less	Il valore presente nel registro al momento della notifica è minore di quello impostato nella proprietà ComparisonValue
LessOrEqual	Il valore presente nel registro al momento della notifica è monore o uguale a quello impostato nella proprietà ComparisonValue
Contains	Il valore presente nel registro al momento della notifica contiene quello impostato nella proprietà ComparisonValue
StartsWith	Il valore presente nel registro al momento della notifica inizia con quello impostato nella proprietà ComparisonValue
EndWith	Il valore presente nel registro al momento della notifica finisce con quello impostato nella proprietà ComparisonValue
AnyChange	Non viene eseguita nessuna comparazione. Le notifiche vengono inviate al cambiamento di qualsiasi valore. Questo, oltre ad essere il valore di default della proprietà ComparisonType, è il comportamento classico in assenza di filtri.

Il secondo concetto molto importante è relativo alle notifiche persistenti. Se stiamo realizzando una applicazione reale che sfrutta le notifiche, al fine di una corretta gestione delle stesse, è lecito supporre che la nostra applicazione resti in esecuzione, altrimenti le notifiche non possono essere gestite. Trattandosi di dispositivi mobili però, tenere l'applicazione sempre in esecuzione può essere più complicato di quanto possa sembrare. Trattandosi di device mobili, siamo soggetti alla carica della batteria, alle risorse di sistema limitate, allo spegnimento accidentale etc. Per far fronte a queste problematiche (e soprattutto alle risorse limitate di sistema), è stato introdotto il concetto di notifiche persistenti. In breve, la nostra applicazione memorizza, nel registro di sistema, una richiesta di notifica su un particolare evento. Quando questo evento si verifica, sarà compito del sistema operativo avviare la nostra applicazione in modo che l'evento in questione possa essere gestito.
Usando codice Managed, eseguire questa operazione è molto semplice: sarà infatti sufficiente creare innanzitutto una istanza di SystemState passando l'informazione relativa allo stato di cui vogliamo ricevere le notifiche, e successivamente richiamare il metodo EnableApplicationLauncher

dell'istanza appena creata. Nella sua forma più semplice, tale metodo accetta come parametro in ingresso una stringa che identifica l'applicazione. Di seguito un esempio di codice che riprende quello dell'esempio precedente, estendendolo in modo da rendere le notifiche persistenti:

```
/// <summary>
/// Chiamante
/// </summary>
private SystemState _incomingCaller;
/// <summary>
/// Identificativo dell'applicazione
/// </summary>
private const string _APPID_ =
                              "ioProgrammo.Article.SNB";

public Form3() {
   InitializeComponent();
   //Istanza di SystemState specializzata sullo stato
                                            da monitorare
   _incomingCaller = new
   SystemState(SystemProperty.PhoneIncomingCaller
                                                  Name);
   //Imposto il tipo di comparazione
   _incomingCaller.ComparisonType =
                              StatusComparisonType.Equal;
   //Imposto il valore da filtrare
   _incomingCaller.ComparisonValue = "Test, Two";
   //La notifica viene resa persistente
   _incomingCaller.EnableApplicationLauncher(
                                              _APPID_ );
   //Evento Change
   _incomingCaller.Changed += new
     ChangeEventHandler(_incomingCaller_Changed);
}
```

In particolare, il metodo _incomingCaller. EnableApplicationLauncher(_APPID_) è quello che si occupa della creazione delle opportune chiavi di registro indispensabili al sistema operativo per individuare ed avviare correttamente l'applicazione (Figura 7).

Come si nota dall'immagine in Figura. 7, tra le varie informazioni automaticamente memorizzate nel registro, emergono:

- Conditional Target: avendo specializzato la notifica filtrando l'Incoming Caller Name, nel registro viene memorizzato il valore in base a cui la notifica deve essere inviata alla nostra applicazione
- Application: è il path completo dell'applicazione da avviare se le condizioni di notifica si verificano
- Value Name: è lo stato a cui siamo registrati per ricevere le notifiche di cambiamento
- HKEY: è il riferimento alla chiave del registro HKEY_LOCAL_MACHINE

Il metodo EnableApplicationLauncher, oltre alla sua forma base, ha due overload molto utili. Il primo accetta, oltre all'ApplicationID, il path dell'applicazione da lanciare. Questo ci da la possibilità, da una applicazione A, di impostare le notifiche persistenti da inviare ad una applicazione B.
Il secondo overload, oltre all'ApplicationId ed al Path, ci permette di inviare una serie di parametri all'applicazione. L'invio di parametri può risultare davvero comodo in scenari in cui l'applicazione deve comportarsi in modo diverso in base al modo in cui viene avviata.

L'AUTORE

Michele Locuratolo è Software Architect per la Mindbox S.r.l. di Capurso (BA), società che si occupa di consulenza e sviluppo software. E' Microsoft .NET MVP nella categoria Visual Developer – Visual C#. E' fondatore di UgiMobile.org, lo User Group Italiano dedicato allo sviluppo su dispositivi mobili. Il suo blog è raggiungibile all'indirizzo http://www.michelelocuratolo.com

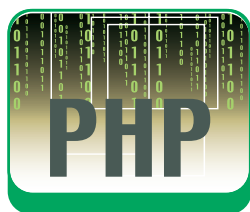
Fig. 7: Il registro di sistema con le informazioni sulle notifiche persistenti.

CONCLUSIONI

State and Notification Broker API è indubbiamente una delle rivoluzioni che riguardano la piattaforma Windows Mobile. Si è passati infatti da un sistema in cui "far girare" le applicazioni, ad un sistema con cui le nostre applicazioni possono integrarsi in modo semplice e, in alcuni casi, possono anche interagire tra loro o con il sistema operativo.
Su questa piattaforma ci sarebbe ancora molto altro da aggiungere in quanto, le sue potenzialità sono davvero grandi. Per approfondirne alcuni aspetti, oltre a consultare la cospicua documentazione on line, è possibile analizzare una applicazione completa, funzionante ed in continuo aggiornamento (ed Open Source) interamente basata su State and Notification Broker. La trovate on line a questo indirizzo: http://www.codeplex.com/MobilePhoneAssistant .

Michele Locuratolo

PROGRAMMAZIONE AVANZATA CON PHP 5

TRA LE MODIFICHE DI MAGGIOR RILIEVO INTRODOTTE DA PHP 5 VI È, SENZA DUBBIO, IL NUOVO MODELLO AD OGGETTI. IN QUEST'ARTICOLO VEDREMO ALCUNE CARATTERISTICHE AVANZATE DELLA PROGRAMMAZIONE ORIENTATA AGLI OGGETTI (OOP) IN PHP 5

PHP 5 (semplicemente PHP da ora in poi) ha introdotto molte migliorie rispetto alla versione precedente, soprattutto per quanto riguarda:

- Programmazione orientata agli oggetti (OOP)
- MySQL
- XML

In questo articolo vedremo alcuni aspetti peculiari di PHP per quanto riguarda la OOP. Non illustreremo, invece, cosa è un'interfaccia o una classe astratta in quanto ci si aspetta che il lettore abbia già familiarità con questi concetti chiave della programmazione orientata agli oggetti. In particolare il seguente articolo tratterà le seguenti tematiche:

- Overloading di alcuni metodi speciali, ossia *__toString, __set, __get e __call.*
- Implementazione di un iteratore. Così facendo possiamo customizzare il comportamento del ciclo *foreach* quando cicla su istanze della nostra classe.
- Ridefinizione dell'operatore [], ossia implementazione dell'interfaccia *ArrayAccess*. Questo ci permetterà di utilizzare l'operatore [] con istanze della nostra classe in modo del tutto trasparente.

REQUISITI

Conoscenze richieste
Medie di OOP e PHP 5.

Software
PHP, Web server

Impegno

Tempo di realizzazione

RIDEFINIZIONE DI METODI SPECIALI

Un metodo che andrebbe ridefinito in ogni classe è, senza alcun dubbio, il buon vecchio *__toString*. I programmatori Java avranno già intuito lo scopo del suddetto metodo. Esso permette di definire la stringa da associare ad un oggetto, quando viene utilizzato in un contesto che coinvolge stringhe. Un esempio renderà tutto più chiaro. Supponiamo di avere una classe, *Person*, con tre proprietà, ossia nome cognome ed indirizzo. Vogliamo ridefinire il metodo *__toString* di modo che quando vengono stampati oggetti di tipo *Person* sia visualizzata una stringa così formata:
Nome Cognome | Indirizzo
Ecco il codice necessario:

```
class Person
{
  private $firstName;
  private $lastName;
  private $address;

  public function setFirstName($firstName)
  {
            $this->firstName = $firstName;
  }

  public function getFirstName()
  {
            return $this->firstName;
  }
  // gli altri getter e setter
  [...]

  public function __toString()
  {
     return $this->firstName . " " . $this->lastName .
                              " | " . $this->address;
   }
}
```

Come si può notare la classe è abbastanza semplice. Contiene le tre proprietà citate ed i metodi getter e setter utilizzati per leggere e scrivere tali proprietà. In più c'è l'overloading di *__toString* il quale fa sì che ogni qual volta che un'istanza della classe *Person* è coinvolta in uno string context sia convertita nella forma: Nome Cognome | Indirizzo. Ad esempio, provate ad utilizzare il seguente codice:

```
$person = new Person();
```

```
$person->setFirstName("Alessandro");
$person->setLastName("Lacava");
$person->setAddress("Via Paco Rinajo, 69 - Milano");

echo "Persona: " . $person;
```

Ciò che otterrete in output è:

```
Persona: Alessandro Lacava | Via Paco Rinajo, 69 –
                                              Milano
```

Come potete vedere l'oggetto *$person* è stato convertito nel formato che abbiamo imposto attraverso il metodo *__toString*.
È importante notare che, dalla versione di PHP 5.2.0, *__toString* è invocato in tutti i contesti che coinvolgono stringhe come ad esempio la concatenazione di stringhe. A tal proposito basta osservare gli effetti prodotti dal seguente pezzo di codice:

```
$person = new Person();
$person->setFirstName("Alessandro");
$person->setLastName("Lacava");
$person->setAddress("Via Paco Rinajo, 69 - Milano");
$otherPerson = new Person();
$otherPerson->setFirstName("David");
$otherPerson->setLastName("Brown");
$otherPerson->setAddress("Via M. Fumeri, 6 -
                                              Como");

echo $person . "<br />" . $otherPerson;
```

L'output prodotto è il seguente:

```
Alessandro Lacava | Via Paco Rinajo, 69 - Milano
David Brown | Via M. Fumeri, 6 - Como
```

PHP 5, però, non si è limitato ad introdurre metodi e concetti già presenti in altri linguaggi, tipo Java, infatti vi sono alcuni metodi "magici" che non hanno corrispettivo nel linguaggio della Sun; sto parlando di *__set, __get* e *__call*. I primi due hanno a che fare con le proprietà di una classe, mentre il secondo con i metodi.
Vediamo come funzionano *__set* e *__get*. Supponiamo di avere sempre la nostra classe *Person*:

```
class Person
{
  private $firstName;
  private $lastName;
  private $address;
  [...]
}
```

Supponiamo, ora, di avere un codice siffatto:

```
echo $person->ghostProperty;
```

La precedente istruzione produce il seguente output:

```
Notice: Undefined property: Person::$ghostProperty
                                    in ... on line 60
```

In pratica abbiamo cercato di leggere il valore di una proprietà che non è stata definita. Fin qui nulla di strano. Il metodo *__get*, tuttavia, ci permette di intercettare questo tipo di chiamate e gestirle nel modo a noi più consono.
Modifichiamo la nostra classe *Person* di modo che includa il seguente overloading di *__get*:

```
public function __get($property)

{
        echo "<br />";
        echo "<b>La proprietà $property non è
                                stata definita</b>";
        echo "<br />";
}
```

Proviamo, ora, a rilanciare il codice precedente, ossia:

```
echo $person->ghostProperty;
```

Stavolta in output otteniamo:

```
La proprietà ghostProperty non è stata definita
```

In pratica *__get* entra in gioco quando sull'oggetto, istanza della classe in cui *__get* è ridefinito, viene invocata una proprietà che non esiste.
L'unico parametro che __get accetta è costituito dal nome della proprietà che si sta cercando di leggere.
Il metodo *__set* funziona in modo complementare, ossia è invocato nel momento in cui si cerca di scrivere una proprietà; in pratica, quando si cerca di assegnargli un valore. Ecco un esempio:

```
public function __set($property, $value)
{
        echo "<br />";
        echo "<b>Non dovresti assegnare il valore
                                $value alla ".
                        "proprietà $property in
                 quanto non è stata definita</b>";
        echo "<br />";
}
```

Proviamo, ora, ad assegnare un valore ad una proprietà non definita:

RISORSE UTILI

PHP Home Page:
http://www.php.net/
Manuale online per PHP 5:
http://it.php.net/manual/en/language.oop5.php

```
$person->ghostProperty = 69;
```

Ecco l'output prodotto dal codice precedente: **Non dovresti assegnare il valore 69 alla proprietà ghostProperty in quanto non è stata definita**

Come avrete immaginato il primo parametro di *__set* è sempre il nome della proprietà che si sta cercando di scrivere, mentre il secondo è il valore che si sta assegnando.
E se invece di intercettare l'accesso a proprietà in lettura/scrittura abbiamo bisogno di gestire in modo customizzato l'invocazione di metodi non definiti? Niente paura, gli ingegneri di PHP hanno pensato anche a questo fornendoci la possibilità di ridefinire il metodo *__call*. Tale metodo è invocato quando si cerca di chiamare un metodo su di una classe in cui non è stato definito. Vediamo prima un esempio banale:

```
function __call($method, $arguments)
{
        echo "Metodo non definito";
}
```

Il codice precedente non fa altro che visualizzare quel messaggio non appena si invoca un metodo non definito di una classe. Tante volte è proprio quello che vogliamo. Il metodo *__call*, tuttavia, può avere applicazioni più articolate ed utili come vedremo più avanti. I parametri che *__call* accetta sono due:

- *$method*: una stringa che rappresenta il nome del metodo che si è cercato di invocare
- *$arguments*: un array contenente gli argomenti passati al metodo. All'indice 0 vi è il primo argomento, all'indice 1 il secondo e così via.

Ma vediamo, ora, un esempio di overloading di *__call* decisamente più utile del precedente. E' buona norma di programmazione orientata agli oggetti definire le proprietà di una classe private e fornire dei metodi di lettura/scrittura (getter e setter) per accedervi. Quando abbiamo classi con molte proprietà, tale operazione può risultare noiosa. Non sarebbe bello avere un metodo che simuli i getter e setter per qualsiasi classe? Con *__call* possiamo fare questo e altro. Vediamo il codice:

```
class Person
{
  private $firstName;
  private $lastName;
  private $address;

 function __call($method, $arguments)
 {
          if(!$this->isValidMethod($method))
          {
                 throw new Exception("Metodo non
                                        supportato");
          }

  $prefix = strtolower(substr($method, 0, 3));
  $property = strtolower(substr($method, 3, 1)) .
                          substr($method, 4);

  if ($prefix == "get")
  {
           if($this->propertyExists($property))
           {
                  return $this->$property;
   }
   else
   {
          throw new Exception("La proprietà
                        $property non esiste");
   }
  }

  if ($prefix == "set")
  {
           if($this->propertyExists($property))
           {
          $this->$property = $arguments[0];
   }
   else
   {
          throw new Exception("La proprietà
                        $property non esiste");
   }
  }
 }
[...]
}
```

Come si può vedere vi sono le tre proprietà già incontrate e, apparentemente, nessun metodo per settare o leggere tali proprietà. Dico apparentemente perché, in realtà, per come abbiamo definito *__call* possiamo leggere e scrivere tutte le proprietà presenti nella classe precedente. Entrando nel dettaglio, il metodo precedente esegue un controllo sulla validità del metodo invocato, attraverso *isValidMethod* che non illustriamo per brevità. In seguito recupera il prefisso (get o set) e la proprietà da leggere/scrivere. Supponendo di invocare il metodo *getFirstName* avremo che *$prefix* e *$property* assumeranno, rispettivamente, i valori get e firstName. In seguito, a seconda il prefisso, la proprietà viene letta o settata; questo dopo averne verificato l'esistenza. Vale la pena dare un'oc-

NOTA

HINT SUI PARAMETRI

PHP non è un linguaggio fortemente tipizzato, vale a dire non c'è bisogno, come in Java ad esempio, di dichiarare il tipo delle variabili. Tuttavia possiamo fornire degli "hint" alle funzioni di modo che accettino parametri solo di un determinato tipo, come abbiamo fatto con la funzione displayPrime la quale accetta solo parametri di tipo PrimeNumber. In realtà PHP, internamente, esegue un controllo tramite l'operatore instanceof. Se tale controllo non va a buon fine lo script fallisce.

chiata al metodo *propertyExist* il quale restituisce true se la proprietà passata come parametro è stata definita e false altrimenti:

```
private function propertyExists($property)
{
        $clazz = new ReflectionClass("Person");
        return $clazz->hasProperty($property);
}
```

Il controllo sull'esistenza o meno della proprietà avviene attraverso l'uso della reflection, altra potente caratteristica di PHP 5. Senza entrare troppo nel dettaglio (la vedremo meglio in un prossimo articolo) il codice precedente "riflettendo" sulla classe *Person* determina se la proprietà *$property* è stata definita o meno. Potete provare il metodo *__call*, definito nella classe precedente, usando il seguente codice:

```
$person = new Person();

$person->setFirstName("Alessandro");
$person->setLastName("Lacava");
$person->setAddress("Via Paco Rinajo, 69 - Milano");

echo "Persona: " . $person;
```

L'output del codice appena visto è il seguente:

```
Persona: Alessandro Lacava | Via Paco Rinajo, 69 -
                                             Milano
```

Questo supposto che la classe contenga il metodo *__toString* ridefinito come visto in precedenza. Come ci aspettavamo le proprietà sono state scritte e lette anche se i corrispettivi setter e getter non sono stati definiti esplicitamente. Magia del metodo *__call*!
Un'altra possibile applicazione del metodo *__call* può essere l'implementazione del delegation pattern. Insomma i limiti di applicazione dei metodi speciali visti in questo paragrafo sono legati solo alla vostra immaginazione.

OVERLOADING DELL'OPERATORE DI INDICIZZAZIONE

PHP permette di ridefinire l'operatore di indicizzazione, ossia le parentesi quadre [], per qualsiasi classe. L'overloading dell'operatore [] è permesso in C++ e C# ma non in Java. A quanto pare i progettisti di PHP 5 hanno, giustamente, cercato di sfruttare i concetti già rodati di linguaggi quali Java e C# ed hanno aggiunto feature particolari come i metodi *__set*, *__get* e *__call*.

Vediamo com'è possibile sfruttare questo potente concetto in PHP, ridefinendo l'operatore di indicizzazione nelle nostra classe *Person*. Supponiamo di avere un insieme record, relativi ad altrettante persone, mantenuti in una tabella di un ipotetico database. Vogliamo effettuare l'overload dell'operatore di indicizzazione di modo da poter scrivere un codice del genere per effettuare un lookup per codice sulla lista di persone:

```
$person = new Person();
echo $person["69lbc"];
```

Dobbiamo scrivere il codice necessario per intercettare la chiamata all'operatore []. Per fare l'overload di tale operatore, in PHP è sufficiente implementare l'interfaccia *ArrayAccess*. Tale interfaccia, come potete vedere in figura 1, espone quattro metodi:

- *offsetExists*: tale metodo accetta in ingresso un indice, numerico o stringa, e restituisce true se l'elemento esiste, false in caso contrario.
- *offsetGet*: accetta anch'esso un indice come unico parametro. E' il metodo invocato, quando si cerca di accedere ad un elemento, in pratica quando si scrive qualcosa del genere:

```
echo $person["69lbc"];
```

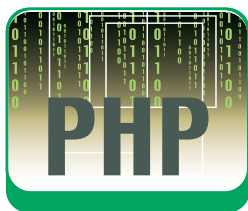
- *offsetSet*: come avrete intuito tale metodo è invocato quando si cerca di assegnare un valore attraverso la notazione utilizzata per gli array. Ad esempio:

```
$person["69lbc"] = $lacava;
```

Nel precedente esempio *$lacava* è un comune oggetto di tipo *Person*. I parametri accettati da questo metodo sono l'onnipresente indice ed il valore da assegnare. Nell'esempio precedente il valore è l'oggetto *$lacava* mentre l'indice è *69lbc*.

- *offsetUnset*: quest'ultimo metodo viene in genere utilizzato per distruggere un elemento

<<interface>> ArrayAccess
offsetExists($index : mixed) : bool offsetGet($index : mixed) : mixed offsetSet($index : mixed,$newValue : mixed) : void offsetUnset($index : mixed) : void

Figura 1: Interfaccia ArrayAccess.

dello pseudo-array. Il parametro che accetta è il solito indice.

Vediamo subito un esempio per la classe *Person*:

```
class Person implements ArrayAccess
{
  private $firstName;
  private $lastName;
  private $address;

  // Reference al DB fittizio
  private $db;

        [...]

        /**
              * Implementazione dei metodi di
                                    ArrayAccess
              */
        function offsetExists($id)
        {
              return $this->db-
                         >userExists($id);
        }

        function offsetGet($id)
        {
              return $this->db->getUser($id);
        }

        function offsetSet($id, $user)
        {
              $this->db->setUser($id, $user);
        }

        function offsetUnset($id)
        {
              $this->db->removeUser($id);
        }
}
```

ARRAY ASSOCIATIVI

E' noto che PHP, anche in versioni precedenti alla 5, offre la possibilità di utilizzare, in modo nativo, array associativi oltre ai classici array a base numerica. In breve un array associativo è un array in cui l'indice è costituito da una stringa invece che da un numero. Ecco un esempio di array associativo:

```
$arr =array(
    "firstName" => "Alessandro",
    "secondName" => "Lacava",
    "address" => "Via Paco
                Rinajo, 69 - Milano"
);
echo $arr["address"];
```

L'output del codice precedente è:

```
Via Paco Rinajo, 69 - Milano
```

Nell'esempio precedente si è supposto che *$db* fosse il reference di una classe DAO (Data Access Object) che ci permette di mappare le righe della tabella Persone ad oggetti di tipo *Person*. Come si può vedere, abbiamo implementato i quattro metodi di *ArrayAccess* di modo da rispettare il contratto esposto da tale interfaccia. Ecco un esempio di codice client che utilizza la classe precedente:

```
$person = new Person();
[...]

// Recupera la persona con id = 69lbc.
// Corrisponde ad invocare il metodo offsetGet
$test = $person["69lbc"];

[...]

// Setta David Brown come utente con id = 96nlm.
// Corrispondere ad invocare il metodo offsetSet
$person["96nlm"] = $brown;
```

```
<<interface>>
Iterator
---------------
rewind() : void
next() : void
valid() : bool
key() : mixed
current() : mixed
```

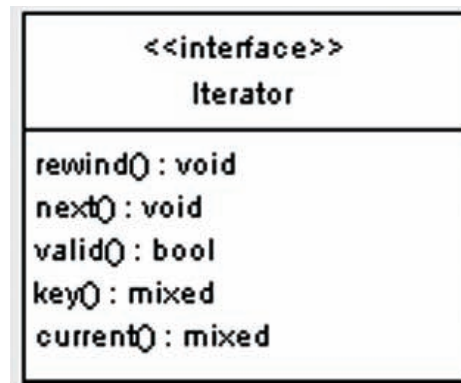



Figura 1: Interfaccia Iterator.

Il codice appena visto utilizza la notazione utilizzata per i comuni array per effettuare operazioni di lettura e scrittura su un ipotetico DB. In pratica il primo esempio corrisponde ad invocare il metodo *offsetGet*, mentre il secondo è come se invocassimo *offsetSet*. In conclusione l'implementazione dell'interfaccia *ArrayAccess* ci permette di customizzare l'utilizzo dell'operatore [] come meglio crediamo.

ITERATORI IN PHP 5

In PHP 5, oltre a poter customizzare la sintassi dedicata agli array, è possibile implementare un iteratore per la propria classe. Per fare ciò è sufficiente implementare l'interfaccia *Iterator* la quale espone cinque metodi, come si evince dalla figura 2.
Così facendo, quando un oggetto della nostra

classe sarà coinvolto all'interno del ciclo *foreach* entreranno in gioco i metodi dell'interfaccia *Iterator*. Lo scopo dei metodi di *Iterator* è il seguente:

- *rewind*: "riavvolge" l'iteratore riportandolo all'inizio della lista.
- *valid*: tale metodo deve restituire un booleano. In particolare deve ritornare true se il valore corrente è valido, false altrimenti. Questo metodo è, in genere, utilizzato per determinare la fine della lista. Corrisponde, a grandi linee, al metodo hasNext dell'interfaccia Iterator di Java.
- *next*: sposta il puntatore dell'iteratore sulla prossima coppia chiave/valore.
- *key*: restituisce la chiave correntemente puntata.
- *current*: restituisce il valore correntemente puntato.

Implementiamo, ora, *Iterator* per uno scopo un po' particolare, ossia trovare i numeri primi compresi in un determinato range. Questo strano impiego di *Iterator* è stato scelto appositamente per dimostrare che è possibile sfruttare questa potente feature di PHP per gli scopi più disparati. Passiamo subito al codice senza perderci in ulteriori chiacchiere:

```
class PrimeNumber implements Iterator
{
        private $start;
        private $end;
        private $current;

        public function __construct($start, $end)
        {
                $this->checkParameters($start,
                                                        $end);

                $this->start = $start;
                $this->end = $end;
        }

        public function rewind()
        {
                $this->current = $this->start;
        }

        public function valid()
        {
                return $this->current <= $this-
                                                        >end;
        }
        public function next()
        {
                $this->current++;
        }
```

```

        public function key()
        {
                return $this->current;
        }

        public function current()
        {
                return $this->isPrime($this-
                                                        >current);
        }

        // Getter e setter
        [...]
}
```

Come si può vedere, abbiamo implementato i cinque metodi di *Iterator*. In più vi sono i getter e setter, un metodo per controllare la validità di *$start* e *$end* ed il metodo *isPrime* che non illustriamo per brevità. In particolare *isPrime* non è stato implementato utilizzando il miglior algoritmo per trovare i numeri primi, visto che non è questo lo scopo del presente articolo. Per i nostri scopi, tuttavia, va più che bene.

Torniamo al nostro codice ed analizziamo l'implementazione dei singoli metodi. Il metodo *rewind* riporta indietro il cursore dell'iteratore facendo puntare l'elemento corrente al primo della lista. Il metodo *valid* restituisce true se il metodo corrente non ha superato la fine (*$end*), false altrimenti. Il metodo *next*, invece, incrementa di 1 l'elemento corrente. Infine vi sono *key* e *current*. Il primo restituisce, semplicemente, l'elemento correntemente puntato, mentre il secondo restituisce true se l'elemento corrente è un numero primo, false in caso contrario.
Vediamo, ora, un codice client che utilizza la classe *PrimeNumber*:

```
$pn = new PrimeNumber(2, 31);
displayPrime($pn);
```

Il codice precedente visualizza i numeri primi compresi tra 2 e 31. In particolare l'output da esso prodotto è il seguente:

```
I numeri primi compresi tra 2 e 31 sono i seguenti:
2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31
```

Vale la pena esaminare la funzione *displayPrime* per vedere come si cicla sull'oggetto di tipo *PrimeNumber*:

```
function displayPrime(PrimeNumber $pn)
{
        echo "I numeri primi compresi tra " .
```

NOTA

TIPI DI DOCUMENTO

**In Windows Presentation Foundation abbiamo a disposizione due tipologie di documenti: FlowDocument e FixedDocument. I primi hanno la particolarità di adattare il contenuto automaticamente in base al device su cui vengono visualizzati riadattando le colonne o i paragrafi.
I FixedDocument, invece, sono documenti con struttura fissa e utili, quindi, a situazioni in cui il layout non deve modificarsi a prescindere dal device. Nel nostro esempio abbiamo utilizzato un fattura che, naturalmente, deve avere un layout ben definito.**

```
                    $pn->getStart(). " e " .
                 $pn->getEnd() . " sono i seguenti:";
    echo "<br />";

    $str = "";
    foreach ($pn as $key => $value)
    {
            if($value)
            {
                    $str .= $key;
                    $str .= ", ";
            }
    }
    $str = substr($str, 0, strlen($str) - 2);
    echo $str;
}
```

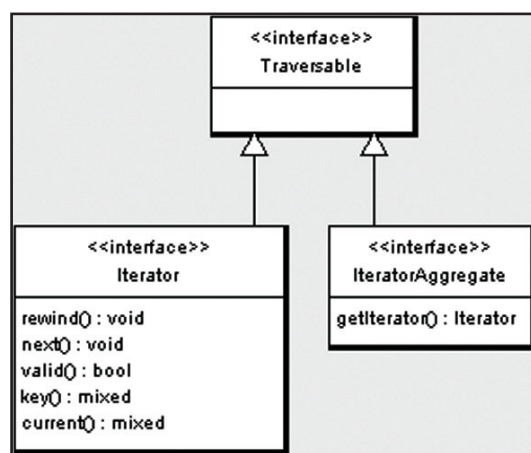



Figura 3: Interfacce Traversable, Iterator e IteratorAggregate

CODICE ALLEGATO

Per provare il codice allegato a quest'articolo avete bisogno di PHP versione 5.2.* ed un Web server.
Il primo lo potete trovare qui: http://www.php.net/
Come Web server io ho utilizzato Apache HTTP Server che potete trovare all'indirizzo: http://httpd.apache.org/

Se utilizzate Windows e volete un'installazione molto semplice di PHP ed Apache Web server consiglio di utilizzare EasyPHP. Esso installa, in un colpo solo, PHP, Apache Web server, MySQL e PHPMyAdmin. Questi ultimi due non vi serviranno per gli esempi di questo articolo, ma vi saranno in ogni modo utili per provare script che utilizzano il database MySQL. L'indirizzo per EasyPHP è il seguente: http://easyphp.org/

Una volta installato ed avviato il server, potete provare gli script allegati utilizzando un URL simile al seguente: http://localhost/ioProgrammo/php5/advanced-oop/object-overloading/PrimeNumber.php

Il precedente URL è valido se avete impostato Apache HTTP Server in ascolto sulla porta 80 (di default è così) e messo il codice allegato sotto il path ioProgrammo/php5/advanced-oop/object-overloading, che a sua volta va inserito sotto la root puntata da localhost. Se avete usato EasyPHP, il path completo sarà qualcosa del genere: C:\Programmi\EasyPHP 2.0b1\www\ioProgrammo\php5\advanced-oop\object-overloading

Come potete vedere, l'aver implementato l'interfaccia *Iterator* ci permette di ciclare sugli oggetti di tipo *PrimeNumber* in modo del tutto trasparente.
L'interfaccia *Iterator*, in realtà, estende un'altra interfaccia, *Traversable*. Tale interfaccia è una "marker interface", ossia non espone metodi da implementare. Lo stesso concetto di marker interface lo ritroviamo in Java con, ad esempio, le interfacce Cloneable e Serializable.
Abbiamo visto come rendere una classe traversable implementando l'interfaccia *Iterator*. In realtà esiste un altro modo per rendere una classe traversabile, ossia implementando l'interfaccia *IteratorAggregate*. Anche quest'ultima estende *Traversable* come si evince dal class diagram di figura 3.
IteratorAggregate espone un unico metodo, vale a dire *getIterator* il quale restituisce un oggetto di tipo *Iterator*.
Utilizzando l'interfaccia *IteratorAggregate*, in congiunzione ad *Iterator* rispettiamo il cosiddetto Single Responsibility Principle (SRP), ossia uno dei principi chiave della programmazione orientata ad oggetti, il quale impone che tra classe e responsabilità da essa assolta vi sia una corrispondenza uno a uno.
Fare il refactoring del codice precedente di modo che usi *IteratorAggregate* assieme ad *Iterator* non è molto complicato. Vi basterà estrarre la logica concernente l'iteratore dalla classe *PrimeNumber* ed in inserirla in una classe a parte che implementa *Iterator*. Inoltre *PrimeNumber* non dovrà più implementare *Iterator*, ma *IteratorAggregate* e, all'interno di *getIterator*, creare e restituire un'istanza della classe che implementa *Iterator*. Vi assicuro che ci vuole più a dirsi che a farsi.

CONCLUSIONI

In questo articolo abbiamo visto alcune peculiarità avanzate di PHP 5. Per chi ha già dimestichezza con linguaggi orientati ad oggetti come Java e C# e conosce un po' di PHP non avrà difficoltà a digerire concetti come interfacce, iteratori e così via.
Per chi, tuttavia, non conosce la programmazione orientata ad oggetti (OOP), ma conosce bene PHP, il mio modesto consiglio è quello di avvicinarvi a questo tipo di programmazione dato che permette di scrivere codice molto elegante e manutenibile. Per lo sviluppo di applicazioni web di dimensioni generose l'OOP vi può rendere la vita molto più semplice.

Alessandro Lacava

ALLA RICERCA DELL'XML PERDUTO

XPATH PERMETTE L'IMPLEMENTAZIONE DI UN MOTORE DI RICERCA ATTRAVERSO UNA INTERFACCIA DI PROGRAMMAZIONE SEMPLICE E VICINA A QUELLA A CUI SI È ABITUATI CON SQL PER I DATABASE RELAZIONALI. IMPARIAMO AD USARLA

XML sapete tutti più o meno cos'è (se così non fosse, prima di leggere l'articolo, è meglio avere un'infarinatura). La prima specifica XML fu pubblicata dal consorzio W3C nel lontano 1998, e sei anni dopo uscì la versione 1.1.
In questo periodo l'industria informatica e non ha adottato XML in lungo e in largo, tanto da farlo divenire probabilmente lo standard per lo scambio dati, fra piattaforme, sistemi, linguaggi, hardware eterogenei. Ogni piattaforma di sviluppo, anche notevolmente differente o addirittura incompatibile, fornisce supporto all'utilizzo di XML. E' il caso di Java, di .NET e così via. Con il diffondersi di XML, sono nati delle tecnologie e altri standard ad esso correlati e su di esso basati. Basti pensare agli schemi XSD, al linguaggio di trasformazione dei fogli di stile XSLT, o ancora ai Web Service. Fra questi standard vi è, ed è argomento di questo articolo, XPath.

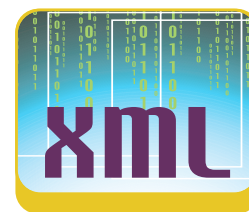
COS'È XPATH

Prima di proseguire, mettendo le mani in pasta, si vuol riepilogare cos'è XPath e a cosa ci servirà.
XPath è un linguaggio per la ricerca di informazioni in un documento XML. XPath permette di navigare attraverso i nodi, gli elementi e gli attributi di tale documento. XPath usa le delle espressioni, dette espressioni path, per selezionare nodi o insiemi di nodi nel documento. Queste espressioni sono molto simili a quelle utilizzate in qualunque linguaggio ed in particolare saranno familiari a chi usa SQL per creare interrogazioni sui database relazionali.
Xpath inoltre include più di 100 funzioni predefinite, che permettono di trattare i diversi tipi di dati, di effettuare confronti, di manipolare nodi e attributi. Infine, sottolineiamo che XPath, come XML, è uno standard approvato dal W3C nel 1999.

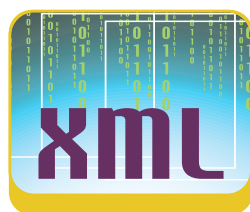
STRUTTURA DI UN DOCUMENTO

Un documento XML è un albero con diversi nodi, e dal punto di vista di XPath ci può essere esattamente una radice ed altri sei tipi di nodi sotto di essa. In pratica quindi ci sono sette possibili tipi di nodi:

- Il nodo radice
- Nodi elemento
- Nodi testuali
- Nodi attributo
- Nodi commento
- Nodi di elaborazione istruzioni
- Nodi namespace

Chi conosce già XML avrà notato la mancanza di alcuni costrutti nell'elenco precedente, per esempio sezioni *CDATA*, dichiarazioni *DTD* e così via. XPath opera infatti su un documento XML dopo che questi elementi sono stati inclusi nel documento stesso. Dunque XPath non può identificare una sezione *CDATA* in un documento.

DATI DI ESEMPIO

Per gli scopi dell'articolo si creerà e si utilizzerà un file XML con la struttura seguente, riportata qui in maniera abbreviata:

```
<articoli>
  <articolo idi"1">
    <titolo>Le query XPath</titolo>
    <autore>Antonio Pelleriti</autore>
    <data>10/09/2007</data>
    <keywords>
      <keyword>xpath</keyword>
      <keyword>query</keyword>
    </keywords>
    <contenuto>
      XPath permette di effettuare delle ricerche
                    all'interno di un documento XML
    </contenuto>
  </articolo>
  <articolo id="2">
    <titolo>Finestre in Python</titolo>
    <autore>Luke Skywalker</autore>
```

```
      <data>24/03/2006</data>
      <keywords>
        <keyword>python</keyword>
      </keywords>
      <contenuto>
        per realizzare delle finestre nel linguaggio python...
      </contenuto>
    </articolo>
</articoli>
```

Il file xml contiene in maniera strutturata diversi articoli ognuno dei quali ha un attributo ID, e dei sottoelementi, titolo, autore, data, parole chiave, ed il contenuto.
Supponiamo di avere un'interfaccia grafica con cui creare tali articolo e salvarli quindi sotto forma di file XML. Potrebbe per esempio essere un motore per la creazione di un blog o di un sito web.
L'esempio

BIBLIOGRAFIA E SITOGRAFIA

XPath Tutorial: il tutorial ufficiale su XPath, linguaggio per trovare informazioni in un documento XML, navigando fra elementi, nodi, attributi:
www.w3schools.com/xpath/default.asp
XML Path Language: Sito ufficale di W3C che descrive il linguaggio e le specifiche
www.w3.org/TR/xpath
.

ESPRESSIONI XPATH

XPath permette di navigare e ricercare i nodi di un documento mediante espressioni costruite con una precisa sintassi.
Per trovare tutti gli articoli contenuti nel precedente file XML, basta una espressione semplicissima come la seguente: *//articolo*. Utilizzando l'esempio del precedente paragrafo verrebbero restituiti due nodi, ognuno dei quali rappresenta un articolo.
L'operatore utilizzato, //, è uno degli operatori standard di XPath. Si vedranno ora i più importanti fra tali operatori, rimandando allo standard (www.w3.org/TR/xpath) per un elenco esaustivo.
L'operatore /, detto child, serve per riferirsi alla radice del documento XML. Per esempio l'espressione /articoli/articolo dice di partire dalla radice del documento, selezionare il nodo articoli e poi selezionare tutti i figli di articoli con nome articolo.
Nell'esempio restituirebbe ancora due nodi.
L'operatore //, già visto, è chiamato operatore ricorsivo discendente. Questo operatore indica di includere tutti i nodi trovati nella ricerca, in maniera ricorsiva. Per esempio scrivendo come già visto //articolo, si parte dalla radice e quando si incontra un nodo con nome articolo, a qualsiasi livello si ha un risultato positivo.
L'operatore wildcard *, trova qualsiasi nodo, per esempio scrivendo /*, si combina l'utilizzo dell'operatore child con l'operatore wildcard, e quindi si trovano tutti i nodi sotto la radice, che nel nostro caso è il nodo articoli. L'espressione /articoli/* invece trova tutti i nodi al di sotto del nodo articoli che si trova a sua volta sotto la radice.
Combinandolo invece con l'operatore ricorsivo discendente, //*, si trovano tutti i nodi a qualsiasi livello del documento.

E' possibile poi utilizzare degli operatori di contesto, come il punto ., che indica il contesto corrente. Ad esempio si potrebbe scrivere del codice che restituisce un certo nodo articolo, e poi a partire da questo si vuole scrivere una espressione ./autore, che restituisce appunto a partire da un nodo articolo il suo autore.
L'operatore .. invece si riferisce al nodo padre: //articoli/articolo/.. restituisce tutti i nodi *articoli*, perché dopo aver navigato in basso fino ad un nodo articolo, si risale su di un livello.
Un operatore molto utilizzato è l'operatore @ di attributo, che permette di selezionare un attributo di un particolare nodo. Per esempio scrivendo //@id si ottengono tutti gli attributi id, nell'esempio verranno restituiti tutti gli id degli articoli presenti.
Nell'esempio applicativo che andremo a realizzare fra poco, cercheremo di effettuare una interrogazione sul file XML, quindi abbiamo bisogno di un operatore che filtri in qualche maniera i nodi del file.
L'operatore filtro è [], che funziona sia con gli attributi che con gli elementi. Scrivendo l'espressione /articoli/articolo[@id=1] verrà restituito il nodo che contiene l'articolo con id pari a 1. E' possibile filtrare anche a livello di elemento, per esempio scrivendo /articoli/articolo[Titolo="titolo"] verranno restituiti i nodi articolo che hanno un nodo articolo con valore "titolo".
È da osservare e da tenere bene a mente che XPath è case sensitive, quindi bisogna fare attenzione nelle espressioni che contengono stringhe.
Naturalmente esistono altri operatori, ed anche funzioni per effettuare operazioni come somme, ricavare sottostringhe, confronti, e così via.
Fra questi, che possono servire per migliorare il servizio di ricerca implementato nell'articolo, citiamo gli operatori logici and e or.
Per trovare per esempio un articolo che abbia come autore Tizio oppure Caio si potrebbe utilizzare la seguente espressione:
//articoli/articolo[autore='Tizio' or autore='Caio']
Rimandiamo il lettore alle specifiche e al tutorial ufficiale, il cui link è riportato in bibliografia.

XPATH E .NET

.NET e la sua Base Class Library mettono a disposizione le funzionalità necessarie a lavorare con XPath ed eseguire ricerche sui file XML come spiegato finora. Per un esempio pratico proveremo a realizzare una applicazione web, ASP.NET, che rappresenta una rivista online, e quindi con articoli, autori, titoli, ecc., contenuti in un file XML come quello utilizzato finora nell'articolo, e che permetta agli utenti di ricercare tali articoli mediante classici filtri del tipo "Ricerca nel titolo" o "Trova articoli contenenti…".
Le classi fondamentali per utilizzare un documento

XML secondo il modello XPath, come vedremo fra breve, sono XPathDocument e XPathNavigator, oltre ad altre classi di utilità che permettono la navigazione fra i nodi del documento stesso.
L'esempio completo contenuto nel cd allegato all'articolo consente anche la creazione di nuovi articoli, e quindi l'aggiornamento del file xml articoli.xml presente nella directory App_Data, utilizzando ancora la classe XPathNavigator.

APPLICAZIONE ASP.NET

Dopo avere creato in Visual Studio 2005 una nuova applicazione Web ASP.NET, bisogna inserire il file XML nella directory speciale App_Data. Si aggiunga poi la directory App_Code, che conterrà il codice di business dell'applicazione ed in questo caso le classi che implementeremo per lavorare più agevolmente con XPath. In particolare la classe XPathUtils, avrà la seguente struttura:

```
public class XPathUtils
{
  private XPathNavigator m_Nav;
  private static XPathDocument m_Doc;
  private XPathNodeIterator m_Iterator;
  public static XPathDocument XMLDoc
  {
    get { return m_Doc; }
    set { m_Doc = value; }
  }
  public XPathUtils(XPathDocument doc)
  {
    try
    {
      m_Doc = doc;
      if (m_Doc != null)
        m_Nav = m_Doc.CreateNavigator();
      else
        throw new Exception("XML Database not
                                        initialized!");
    }
    catch
    {
      throw;
    }
  }
  public XPathNodeIterator
              ExecuteXPathQuery(string expression)
  {
    try
    {
      if (String.IsNullOrEmpty(expression))
        return null;
      m_Iterator = m_Nav.Select(expression);
      return m_Iterator;
    }
    catch(Exception ex)
    {
      throw ex;
    }
  }
```

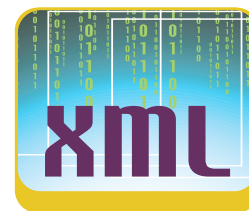
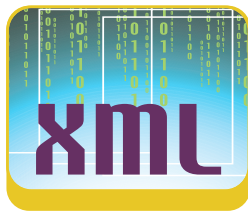
Tale classe ci permetterà di aprire un documento XML ed eseguire delle query secondo la sintassi esposta parlando di XPath.
Aggiungiamo ora una pagina default.aspx, nella quale visualizzeremo i risultati di ricerca, e con i controlli necessari ad impostare dei filtri. Per questo semplice esempio permetteremo la ricerca nel campo Titolo e Contenuto, lasciando al lettore il compito e la possibilità di espandere le opzioni di ricerca e le modalità. Per esempio suggeriamo di provare a implementare una ricerca del tipo "Autore comincia per" oppure "Articoli dal gg/mm/aa al gg/mm/aa".

OBJECTDATASOURCE E DATALIST

Per visualizzare i risultati utilizzeremo un controllo standard DataList, e come sorgente dati invece un ObjectDataSource. Per utilizzare quest'ultimo è necessario definire una classe che contenga le informazioni da visualizzare. In questo caso la classe si chiamerà SearchResults e restituirà gli elementi da visualizzare per mezzo del metodo GetArticoli:

```
public class SearchResults
{
  public static List<Articolo> articoliTrovati=new
                                    List<Articolo>();
  public SearchResults(List<Articolo> art)
  {
      articoliTrovati = art;
  }
  public List<Articolo> GetArticoli()
  {
    return articoliTrovati;
  }
}
```

Il metodo GetArticoli restituisce una List di oggetti Articolo, la cui struttura corrisponde a quella del nodo Articolo nell'XML di esempio, quindi è la seguente, escludendo le proprietà:

```
public class Articolo
{
  private int id;
  private string titolo;
  private string autore;
  private DateTime data;
  private List<string> keywords;
```

```
    private string contenuto;
            ...
}
```

La DataList è configurata in maniera da utilizzare il controllo ObjectDataSource in questo modo:

```
<asp:DataList ID="DataList1" runat="server"
                          DataSourceID="xpathDataSource">
        <ItemTemplate>
            Autore:
            <asp:Label ID="AutoreLabel"
            runat="server" Text='<%# Eval("Autore") %>'>
</asp:Label><br />
            Titolo:
            <asp:Label ID="TitoloLabel" runat="server"
                         Text='<%# Eval("Titolo") %>'>
</asp:Label><br />
            Data:
            <asp:Label ID="DataLabel" runat="server"
                         Text='<%# Eval("Data") %>'>
</asp:Label><br />
            Keywords:
            <asp:Label ID="KeywordsLabel"
        runat="server" Text='<%# Eval("Keywords") %>'>
</asp:Label><br />
            <asp:HyperLink ID="hyp1" runat="server"
        NavigateUrl='<%# "Default.aspx?id="+ Eval("Id")
                                                     %>'>
Leggi
</asp:HyperLink>
            </ItemTemplate>

                </asp:DataList>
<asp:ObjectDataSource ID="xpathDataSource"
                runat="server" SelectMethod="GetArticoli"
            TypeName="SearchResults"
                OnSelecting="xpathDataSource_Selecting">
</asp:ObjectDataSource>
```

Si noti che l'ObjectDataSource utilizza come tipo di dati SearchResults, invocando il metodo GetArticoli, allo scatenarsi dell'evento OnSelecting.

PAGINA DI RICERCA

La pagina asp.net dalla quale effettueremo le ricerche e che visualizzerà i risultati e mostrata in figura 1. E' possibile ricercare all'interno dei nodi Titolo e Contenuto, ma siete liberi di espandere la ricerca anche agli altri nodi.

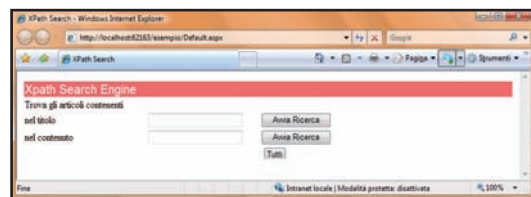



Fig. 1: La pagina di ricerca e visualizzazione

Cliccando su uno dei pulsanti di ricerca, si scatena come detto l'evento OnSelecting, invocando il metodo Select del DataSource. Per esempio così:

```
protected void btRicercaContenuto_Click(object
                                   sender, EventArgs e)
{
    xpathExpression =
  String.Format("//articolo/contenuto[contains(.,\'{0}\'
                              )]", txtTitleFilter.Text);
    xpathDataSource.Select();
    DataList1.DataBind();
}
```

L'espressione Xpath qui costruita viene utilizzata poi nel gestore dell'evento OnSelecting, che a sua volta crea e utilizza l'oggetto XPathUtils che si occupa della ricerca vera e propria:

```
public void xpathDataSource_Selecting(object
        sender, ObjectDataSourceSelectingEventArgs e)
  {
    XPathDocument document = new
                                     XPathDocument(
Server.MapPath("~/App_Data/articoli.xml"));
    XPathUtils xpath = new XPathUtils(document);
    results =
            xpath.SearchArticoli(xpathExpression);
    if (results != null)
        SearchResults.articoliTrovati =
                                 results.GetArticoli();
    else SearchResults.articoliTrovati = null;
  }
```

Il metodo di XPathUtils che svolge il lavoro dietro le quinte è in questo caso SearchArticoli.

```
public SearchResults SearchArticoli(string expression)
{
    try
    {
        if (String.IsNullOrEmpty(expression))
            return null;
        m_Iterator = m_Nav.Select(expression);
        List<Articolo> articoli = new List<Articolo>();
        Articolo art;
        while (m_Iterator.MoveNext())
        {
            art = new Articolo();
             m_Iterator.Current.MoveToParent();
            art.Id =
        Convert.ToInt32(m_Iterator.Current.GetAttribute
                                       ("id", ""));//id
            m_Iterator.Current.MoveToFirstChild();
            art.Titolo = m_Iterator.Current.Value;
            m_Iterator.Current.MoveToNext();
```

```
            art.Autore = m_Iterator.Current.Value;
            m_Iterator.Current.MoveToNext();
            art.Data =
               DateTime.Parse(m_Iterator.Current.Value);
            m_Iterator.Current.MoveToNext();
            if (m_Iterator.Current.MoveToFirstChild())
            {
               do
               {
art.KeywordsList.Add(m_Iterator.Current.Value);
               }
               while (m_Iterator.Current.MoveToNext());
               m_Iterator.Current.MoveToParent();
            }
            articoli.Add(art);
         }
         SearchResults res = newSearchResults(articoli);
         return res;
      }
      catch(Exception ex)
      {
         throw;
      }
}
```

Mediante il metodo Select dell'oggetto XPathNavigator viene effettuate la query XPath vera e propria e restituito un XPathNodeIterator. Se sono stati trovati nodi "articolo" che soddisfano i criteri di ricerca, il metodo MoveNext continuerà a restituire true per tante volte quanti sono gli articoli. Per ogni ciclo viene poi costruito un oggetto articolo, muovendosi fra nodi, e sottonodi.
Gli oggetti articolo così costruiti vengono inseriti quindi in un oggetto SearchResults, che sarà quello utilizzato dal controllo ObjectDataSource e quindi dal DataList.

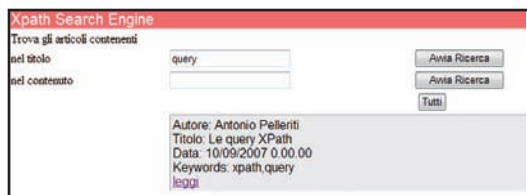



Fig. 2: risultati di una ricerca

Cliccando su uno dei link leggi, verrà effettuata una nuova query XPath, stavolta utilizzando come parametro solo l'id, che viene passato alla pagina tramite QueryString:

```
xpathExpression
          =string.Format("/articoli/articolo[@id={0}]",id);
```

Come già visto verrà costruito un oggetto Articolo, visualizzato poi all'interno di uno User Control dedicato.

INSERIMENTO ARTICOLI

Per inserire un articolo all'interno del file XML, può essere ancora usato XPath. Si veda il codice allegato per un maggiore dettaglio. Il seguente metodo mostra una traccia di come è possibile effettuare un inserimento, ma si noti che dovrebbe essere calcolato un id univoco per l'articolo da aggiungere (cosa fattibile con una query che ricavi l'ultimo id per esempio):

```
XmlDocument document = new XmlDocument();
    document.Load(Server.MapPath("~/App_Data/articoli.
                                            xml"));
XPathNavigator navigator =
                          document.CreateNavigator();
if (navigator.CanEdit)
{
        navigator.MoveToChild("articoli", "");
        XmlWriter xtw = navigator.AppendChild();

        xtw.WriteStartElement("articolo");
        xtw.WriteAttributeString("id", "X");
        xtw.WriteElementString("autore",
                                      txtAutore.Text);
        xtw.WriteElementString("data",
          Calendar1.SelectedDate.ToShortDateString());
        xtw.WriteStartElement("keywords");
        foreach(string kw in txtKeywords.Text.Split(';'))
            xtw.WriteElementString("keyword", kw);
        xtw.WriteEndElement();

        xtw.WriteElementString("contenuto",
                                   txtContenuto.Text);

        xtw.WriteEndElement();

        xtw.Close();
        xtw.Flush();

        navigator.MoveToRoot();
document.Save(Server.MapPath("~/App_Data/articoli.
                                            xml"));
        return;
}
```

CONCLUSIONI

Nell'articolo si è visto come si possono effettuare delle interrogazioni, dei filtraggi e delle ricerche su un file XML utilizzando il modello Xpath, standard W3C. In questa maniera, in casi semplici ma non solo questi, è possibile utilizzare un file XML come vero e proprio database, senza dover ricorrere ad installazione di server e dover espandere o sostituire hardware per avere maggiore potenza di calcolo.

Antonio Pelleriti

L'AUTORE

Antonio Pelleriti, ingegnere informatico, sviluppa software da più di dieci anni e si occupa di .NET sin dalla prima versione Beta. È chief software architect di DynamiCode s.r.l., Software Factory in cui progetta e svviluppa soluzioni custom ed in outsourcing (www.dynamicode.it). Può essere contattato per suggerimenti, critiche o chiarimenti all'indirizzo e-mail antonio.pelleriti@dotnetarchitects.it

SPAMMER ADDIO CON IL CAPTCHA

RICONOSCI GLI UTENI REALI DAI BOT. SOLO CHI È IN GRADO DI IMMETTERE MANUALMENTE I CARATTERI CORRISPONDENTI A QUELLI GENERATI IN MANIERA CASUALE DAL TUO SITO È UNA PERSONA, TUTTI GLI ALTRI SONO GLI ODIOSI BOT USATI DAGLI SPAMMER!

In questo articolo parleremo di CAPCTHA. Niente paura, nonostante il termine sia inusuale si tratta di qualcosa che incontriamo spesso in giro per il Web.
Diciamo innanzitutto che CAPCTHA non è altro che uno dei soliti acronimi che si incontrano nel mondo dell'informatica e sta per *Completely Automated Public Turing test to tell Computers and Humans Apart* (Test di Turing pubblico e completamente automatico per distinguere computer e umani) in pratica si intende un programma in grado di distinguere tra persone e computer.
Un tipico test CAPTCHA è quello in cui si richiede ad un utente di riscrivere una sequenza di lettere o numeri che appaiono distorti o offuscati sullo schermo. Ne troviamo numerosi esempi in siti che offrono servizi di controllo come il WHOIS (vedi figura 1).

Fig. 1: CAPTCHA presente sul WHOIS di www.nic.it

L'intento di questo tipo di test è evitare che dei programmi (BOT) possano emulare il comportamento dell'utente (come il riempimento e l'invio di una Form) arrivando così o a bypassare il sito che offre un dato servizio o ad intasarlo di richieste che, proprio perché generate da computer, possono essere numerosissime.
Per essere ancora più chiari facciamo un esempio. Avete sviluppato una super-applicazione web che offre agli utenti di calcolare l'affinità di coppia e collocate l'applicazione nel vostro sito www.coppiefelici.com incrementando notevolmente i contatti. Un bel giorno però il sito concorrente, www.coppiecontente.com, stanco di perdere utenti studia un programma (BOT) che presenta al navigatore una form di inserimento dati, il programma prende questi dati, li invia con una Request al vostro server e intercetta la risposta rinviandola come output all'utente che così resta nel sito www.coppiecontente.com e non sospetta minimamente che il servizio è invece offerto da www.coppiefelici.com.
Il rimedio in questi casi è mettere un bel CAPTCHA nella Form, ovvero far apparire un'immagine contenente una sequenza di caratteri distorti che l'utente deve digitare per poter accedere al servizio. In questo modo un umano non avrà difficoltà a leggere e riscrivere quanto legge nell'immagine, mentre per un programma questo sarà molto più difficile.

CAPTCHA IN ASP.NET

Fin qui tutto bene, ma come fare ad inserire un test CAPTCHA nelle pagine web che utilizzano la tecnologia ASP.NET?
ASP.NET non implementa i test CAPTCHA tra i controlli standard, tuttavia è disponibile un ottimo controllo Open Source scaricabile dal sito www.guru4.net.
Il controllo è scritto in C# e fornito anche dei sorgenti.
Vediamo quindi, nel dettaglio, l'implementazione e il funzionamento di questo controllo.
Per prima cosa scarichiamo il progetto dal sito www.guru4.net e compiliamolo in Visual Studio 2005. L'output del progetto è una libreria che si chiama *GURU4.net.Web.Controls.*

CaptchaLibrary.dll.
Per avere a disposizione i controlli, in maniera visuale, nell'IDE di Visual Studio cliccare con il tasto destro nella "casella degli strumenti" (Toolbox) e scegliere l'opzione "scegli elementi".
Nella finestra di dialogo (figura 2) che si aprirà fare clic sul bottone "sfoglia" e scegliere il percorso su disco della libreria.
Una volta aggiunta la libreria nella casella degli strumenti avremo due nuovi elementi :

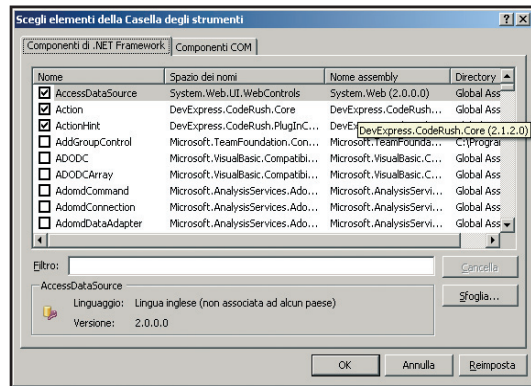



Fig. 2: finestra di dialogo scegli elementi della casella degli strumenti

- VisualCaptcha – che si riferisce all'immagine di Test
- VisualCaptchaValidator – che gestisce la validazione dell'input dell'utente

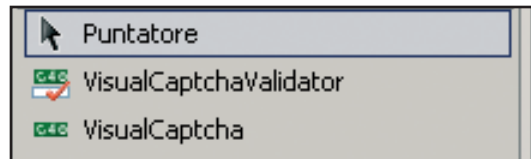



Fig. 3: i controlli aggiunti alla Toolbox

A questo punto dovremo effettuare una piccola modifica al web.config del sito, sotto <system.web> inseriamo:

```
<httpHandlers>
    <add verb="GET" path="visualcaptcha.axd"
              type="GURU4.net.Web.Controls.
        CaptchaLibrary.VisualCaptchaHandler"
    />
</httpHandlers>
```

Ovviamente se la sezione *<httpHandlers>* esiste già ci limiteremo ad inserire l'*<add>*.
Questa operazione abilita l'handler della libreria che crea dinamicamente l'immagine CAPTCHA, è quindi fondamentale compierla correttamente per la riuscita dell'operazione.
Creiamo quindi una nuova pagina *.aspx* (o apriamone una esistente) ed inseriamo, in un *panel* (*Panel1*):

- un controllo *VisualCaptcha* (lo chiamiamo *ImgCaptcha*)
- una *textbox* (*TxtBox*) per la convalida del testo
- un controllo *VisualCaptchaValidator* (*CaptchaValidator*) che gestirà la validazione dell'input
- un bottone (*BtnTest*) per eseguire il test

Oltre a ciò inseriamo poi un altro panel (Panel2) inizialmente non visibile (Visible=false) che ospita il messaggio di superamento del test e un link per ricaricare la pagina.
Il tutto si presenterà come in figura 4.
A questo punto occorre procedere con qualche piccolo settaggio dei controlli.
Il controllo *VisualCaptcha* dispone di alcune proprietà per regolare il tempo di validità della stringa CAPTCHA (proprietà Expiration), il numero dei caratteri della stringa (ChallengeTextLength) e l'url usato per l'handler (RenderUrl) che deve corrispondere a quello indicato nel web.config (nel nostro caso *visualcaptcha.axd*). Qui possiamo

Fig. 4: il layout della pagina di test

LUCI ED OMBRE SU CAPTCHA

I test CAPTCHA sono generalmente considerati una barriera contro gli spammer o altre forme di abuso.
Intorno alle tecniche di test CAPTCHA, ed in particolare al meccanismo di ripetizione di testo distorto contenuto in immagini, c'è però un acceso dibattito che ne contesta i limiti all'accessibilità dei servizi da parte di non vedenti o ipo-vedenti, in effetti questo è un problema visto che anche il W3C con il documento http://www.w3.org/TR/turingtest pone la questione in termini critici. Oltre all'accessibilità c'è poi un problema di reale efficacia del test, visto che cominciano ad vedersi i primi software che riescono a violare i test CAPTCHA.
Rimando all'interessante voce su wikipedia :
http://it.wikipedia.org/wiki/CAPTCHA

IL CODICE ALLEGATO

Gli esempi nell'articolo sono in Visual Basic; per chi fosse interessato, nel CD allegato è presente tutto il codice sorgente anche in C#.

comunque lasciare impostate le proprietà sui valori di default.

Per il controllo *VisualCaptchaValidator* occorre invece impostare:

- **AssociatedVisualCaptchaControlId** – ovvero l'id del controllo VisualCaptcha a cui è associato il Validator (nel nostro caso ImgCaptcha)
- **ControlToValidate** – l'id del controllo da validare (la textbox TxtBox)

Infine occorre inserire il codice da eseguire

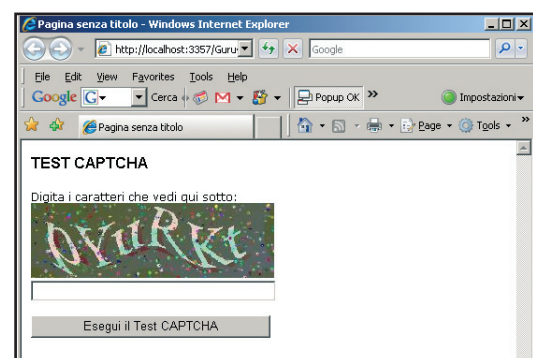



Fig. 5: Il test CAPTCHA in fase iniziale

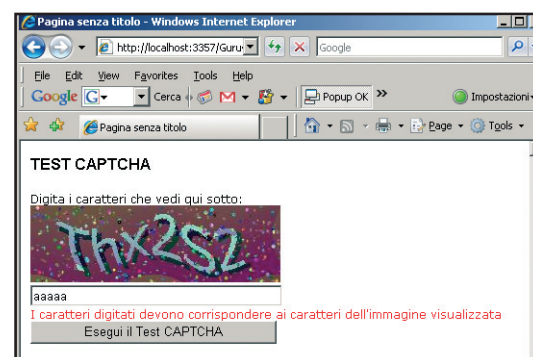



Fig. 6: Il test CAPTCHA fallito

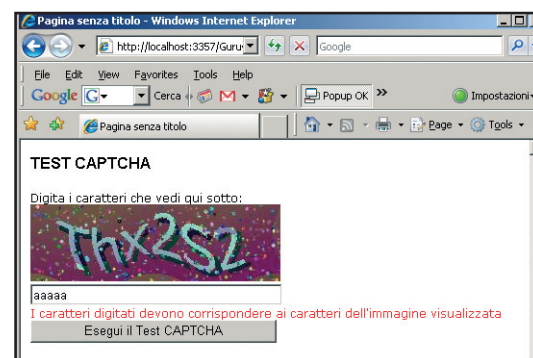



Fig. 7: Il test CAPTCHA superato

sul click sul bottone, ovvero semplicemente:

```
Protected Sub BtnTest_Click(ByVal sender As
```

```
        Object, ByVal e As System.EventArgs)
    If Page.IsValid Then
        Panel1.Visible = False
        Panel2.Visible = True
    End If
End Sub
```

Tutto qui! Se tutto è andato bene nel browser potremo vedere il nostro test CAPTCHA in azione (figure 5-6-7).
A questo punto potremmo anche avere finito no? C'è il controllo, l' implementazione, cosa manca?
Beh, chi sviluppa in ASP.NET seguendo il sistema "standard" degli eventi e dei controlli server in effetti può anche fermarsi qui.
Alcuni invece, come il sottoscritto, ritengono che il sistema degli eventi e dei controlli server sia "artificioso" e tenda a produrre codice HTML inutilmente pesante (oltre ad avere altri svantaggi), e quindi tendono a sviluppare anche in ASP.NET in modo simile ad ASP o PHP (seguendo una logica Request/Response).

PERSONALIZZAZIONE DEL CODICE SORGENTE

Non utilizzando, di norma, controlli server, ed un modello ad eventi collegato ad essi, non sarebbe quindi possibile utilizzare il controllo di cui abbiamo parlato.
In realtà però è possibile, con qualche piccola modifica al codice sorgente, utilizzare le funzionalità esposte dal controllo CAPTCHA anche senza ... nessun controllo!
Vediamo prima di tutto di comprendere meglio la struttura del progetto del CAPTCHA Control.
Se prendiamo il codice HTML generato da una pagina dov'è presente un controllo *VisualCaptcha* scopriremo, tra l'altro, qualcosa si molto simile a:

```
<img style="border-width: 0px; height: 80px;
                                    width: 260px;"
                              src="visualcaptcha.axd?
      id=52d1372f81c3421ca0bf55133922dd09"/>
```

Ovvero c'è un'immagine (quella che rappresenta il testo distorto) che non è un file statico presente su disco, ma un indirizzo ad un'handler (*visualcaptcha.axd*).
Ricordiamo che un indirizzo di tipo "*.axd*" (così come anche gli "*.ashx*") non deve necessariamente corrispondere ad un file fisico su disco, nel *web.config* infatti si può assegnare

l'indirizzo ad una classe che implementa IHTTPHandler che può essere definita in una libreria. E questo è appunto il nostro caso. Ricordate quanto abbiamo aggiunto al *web.config*?

```
<add verb="GET" path="visualcaptcha.axd"
                          type="GURU4.net.Web.
    Controls.CaptchaLibrary.VisualCaptchaHandler"
                                              />
```

Ebbene, da ciò se ne deduce che l'indirizzo "*visualcaptcha.axd*" viene gestito dalla classe *GURU4.net.Web.Controls.CaptchaLibrary.VisualCaptchaHandler* non resta quindi che andare a cercare nei sorgenti forniti insieme al controllo la classe corrispondente.
La troviamo nel file di nome "*VisualCaptchaHandler.cs*". Studiando il codice si comprende il funzionamento interno del controllo (spiegato anche nel sito):

- Le informazioni di rappresentazione dell'immagine per la verifica (dimensioni, numero di caratteri da visualizzare e stile) vengono impostate nel VisualCaptcha control inserito nella pagina.
- Eseguendo la pagina viene generato un codice casuale e le informazioni vengono salvate nella cache di ASP.NET, con la scadenza definita dal tempo limite per la soluzione del test.
- L'handler, recuperando le informazioni dalla cache, visualizzerà l'immagine per la prova e, all'invio della richiesta (postBack) la risposta immessa dall'utente verrà verificata tramite il VisualCaptchaValidator control.

In sintesi, il controllo VisualCaptcha interagisce con la pagina ospite collegando ad essa le informazioni contenute nell'immagine (sequenza di caratteri) utilizzando l'oggetto cache di ASP.NET.
È evidente però, per chi vuole seguire un altro modello di sviluppo, questa soluzione è troppo legata all'utilizzo di controlli e quindi inutilizzabile.
Quello che ci vorrebbe è un gestore (HttpHandler) che, in pratica, faccia la stessa cosa di "*VisualCaptchaHandler.cs*" (ovvero disegnare un'immagine con una sequenza di caratteri distorti) che scriva anch'esso la stringa di Test nella cache di ASP.NET (collegandola a una chiave univoca per associarla ad una specifica richiesta), ma che sia in grado di accettare i parametri come Querystring. Cioè, per ottenere un'immagine CAPTCHA di 300x80 con 5 caratteri, dovrebbe essere scrivere:

```
<img src="visualcaptcha.axd?
                    w=300&h=80&l=5"/>
```

anziché dover passare tali parametri necessariamente attraverso le proprietà di un controllo.
A questo punto potremmo essere tentati anche di scrivere da zero un nostro Handler per disegnare l'immagine CAPTCHA, tuttavia il codice della classe *VisualCaptchaHandler* fa già per il 90% al caso nostro, perché dunque "inventare l'acqua calda"?
Rispetto alle nostre esigenze, l'unico problema della classe *VisualCaptchaHandler* è che legge e scrive le proprietà da ed in un oggetto, *VisualCaptchaMetaData*, quindi una soluzione potrebbe essere quella di creare, nel progetto *VisualCaptcha*, una nuova classe (che chiameremo *BasicCaptchaHandler* ad indicare che non è collegata ai controlli).
In questa classe copieremo tutto il codice contenuto nel file *VisualCaptchaHandler.cs*
La classe disporrà di un metodo per la lettura delle variabili dalla QueryString con la possibilità di indicare un valore di default:

```
private string RequestVar(string name, string
                                defaultValue)
{
        string v = request.QueryString[name];
        if (String.IsNullOrEmpty(v)) return
                                defaultValue;
        else return v;
}
```

Poi doteremo la classe di proprietà che derivano appunto dalla lettura dalla QueryString (proprietà che in *VisualCaptchaHandler* erano impostate tramite il controllo):

```
private int ChallengeTextLength
{
      get {
          return int.Parse(RequestVar("l", "6"));
       }
}

private int Width
{
        get { return
              int.Parse(RequestVar("w","180"));
        }
}
private int Height
{
```

SYSTEM.DRAWING

Il namespace System.Drawing fornisce una serie di classi per creare o modificare immagini di diverso formato. Utilizzando queste classi è possibile creare grafici con dati recuperati da database, oppure modificare immagini preesistenti. Ulteriori informazioni e la lista completa di classi e metodi sono disponibili all'indirizzo: http://msdn2.microsoft.com/it-it/library/system.drawing(VS.80).aspx

```
    get {
        return int.Parse(RequestVar("h", "80"));
    }
}
private int Expiration {
    get {
      return int.Parse(RequestVar("e", "1800"));
    }
}
```

Il cambiamento principale rispetto al codice di *VisualCaptchaHandler* l'avremo sul metodo *ProcessRequest*. Nella classe *VisualCaptcha Handler* il corpo del metodo era:

```
Guid id = new
 Guid(context.Request.QueryString[VisualCaptcha.
                                          IdKey]);
  VisualCaptchaMetaData metadata =
      VisualCaptcha.GetVisualCaptchaMetaData(id);
  context.Response.Clear();
  context.Response.ContentType = "img/jpeg";
  using (Bitmap bmp =
              GenerateImage(metadata.Text, new
         Size(metadata.Width, metadata.Height)))
  bmp.Save(context.Response.OutputStream,
ImageFormat.Jpeg);
```

Nella nostra classe (*BasicCaptchaHandler*) invece il corpo del metodo diventa:

```
Guid id =new Guid(RequestVar("id",""));

string Text =
        GetRandomCode(this.ChallengeTextLength);
context.Response.Clear();
context.Response.ContentType = "image/jpeg";
context.Cache.Insert(id.ToString(), Text, null,
        DateTime.Now.AddSeconds(this.Expiration),
 System.Web.Caching.Cache.NoSlidingExpiration);
using (Bitmap bmp = GenerateImage(Text, new
                   Size(this.Width, this.Height)))
bmp.Save(context.Response.OutputStream,
                              ImageFormat.Jpeg);
```

Come differenze principali abbiamo :

- Le proprietà derivano dalla QueryString e non dal controllo.
- Nella cache ASP.NET non viene inserito l'oggetto VisualCaptchaMetaData ma soltanto il valore di testo generato associato al'ID.
- L'inserimento nella cache ASP.NET avviene contestualmente alla produzione dell'immagine di output.

Bene, a questo punto possiamo ricompilare la libreria e modificare il *web.config* per aggiungere il nuovo Handler:

```
<add verb="GET" path="captcha.axd"
                  type="GURU4.net.Web.Controls.
           CaptchaLibrary.BasicCaptchaHandler"/>
```

Bene, per testare il funzionamento della nuova classe BasicCaptchaHandler scriviamo una semplice pagina .aspx che fa esattamente ciò che abbiamo visto in figura 5-6 e 7, ma evitando l'uso di web controls, questo sarà il layout:

```
<form method="post" action="Default.aspx">
   <div id="panelTest" runat="server"
                                  visible="true">
      <h2>TEST CAPTCHA
          <%=RecuperaTestoDaCache()%></h2>
       Digita i caratteri che vedi qui sotto:<br />
      <img src="captcha.axd?id=<%= CaptchaId
                          %>&w=259" alt=""/>
      <input type="hidden"
              name="PreviousCaptchaId" value="
                         <%= CaptchaId %>" />
      <div>
      <input type="Text"
      style="width:256px;margin-top:1px;margin-
         bottom:10px;" name="txtCaptcha"/>
      </div>
       <input type="submit" style="width:256px"
              name="Esegui il Test CAPTCHA"/>
   </div>
   <div id="panelFailed" runat="server"
                                 visible="false">
      I caratteri digitati devono corrispondere ai
             caratteri dell'immagine visualizzata!
   </div>
   <div id="panelSuccess" runat="server"
                                 visible="false">
        <h2>HAI SUPERATO IL TEST
                                   CAPTCHA!</h2>
         <a href="Default.aspx">
                  Ripeti il test</a>
   </div>
</form>
```

La logica di esecuzione anche qui è semplice ed è contenuta sull'evento Load (ovvero al caricamento della pagina). In pratica si controlla che il caricamento non avvenga mediante un POST dell'utente, se è così si recupera l'id associato all'immagine CAPTCHA visualizzata in precedenza e si cerca nella Cache ASP.NET la stringa rappresentata nell'immagine; infine si confronta con quella inviata dell'utente e, se corrispondono, si nasconde il <DIV> *panelTest* e si rende visibile *panelSuccess*, se invece c'è errore ovviamente si rende visibile *panelFailed*.
Tutto questo si traduce nel codice:

```
Public CaptchaId As String =
                Guid.NewGuid().ToString()

Public Function RequestVar(ByVal
        name As String,  ByVal defaultValue
                        As String) As String

    Dim v As String =
                Request.QueryString(name)
    If String.IsNullOrEmpty(v) Then
        v = defaultValue
    End If
    Return v
End Function

Private Function RecuperaTestoDaCache() As
                                    String
    Dim PreviousCaptchaId As String =
            Request.Form("PreviousCaptchaId")
    If Not
        String.IsNullOrEmpty(PreviousCaptchaId)
                                       Then
        If Cache.Item(PreviousCaptchaId) IsNot
                               Nothing Then
            Return
         Cache.Item(PreviousCaptchaId).ToString
        End If
    End If
    Return String.Empty
End Function

Protected Sub Page_Load(ByVal sender As Object,
            ByVal e As System.EventArgs)
    Dim txtCaptcha As String =
                Request.Form("txtCaptcha")
    Dim compareText As String =
                RecuperaTestoDaCache()
    If Request.Form.Count > 0 Then
        'POST
        If Not String.IsNullOrEmpty(txtCaptcha)
   AndAlso Not String.IsNullOrEmpty(compareText)
        AndAlso txtCaptcha.Equals(compareText,
                          StringComparison.
                InvariantCultureIgnoreCase) Then
            Me.panelTest.Visible = False
            Me.panelFailed.Visible = False
            Me.panelSuccess.Visible = True
        Else
            Me.panelTest.Visible = True
            Me.panelFailed.Visible = True
            Me.panelSuccess.Visible = False
        End If
    Else
        Me.panelTest.Visible = True
        Me.panelFailed.Visible = False
        Me.panelSuccess.Visible = False
    End If
End Sub
```

A questo punto, nel browser avremo un risultato del tutto identico al nostro test che utilizzava il web control.
Già - direte voi – ma se il risultato lo stesso perché allora non usare semplicemente il web control e non complicarsi la vita con handler ed altra roba del genere?
Presto detto! Apriamo le due pagine (quella che utilizza i Web Control e l'altra che si limita al classico meccanismo Request/Response) in Firefox con l'estensione Firebug abilitata e controlliamo i risultati riportati in figura 8 e 9.

***Fig. 8:** tempi di caricamento della pagina con il web control*

***Fig. 9:** tempi di caricamento della pagina con meccanismo Request/Response*

CONCLUSIONI

Come potete notare il sistema di programmazione standard ASP.NET (Web Control ecc...) è più di due volte più lento (594 ms contro 234) rispetto al classico meccanismo Request/ Response.
Mi sembra che il raddoppio delle prestazioni sia un argomento più che valido per impegnarsi un po' di più in fase di sviluppo a cercare soluzioni che non necessariamente devono essere quelle c.d. "standard". D'altra parte è anche vero che la grande versatilità di ASP.NET consente appunto di scegliere il metodo più idoneo per le proprie esigenze. Se le prestazioni per voi non sono un problema potete sempre scegliere sempre il metodo classico. Se inveve siete alla ricerca delle performance senza dubbio qualche riga d condice in più può aiutare

Francesco Smelzo

L'AUTORE

**Francesco Smelzo è specializzato nello sviluppo in ambiente Windows con particolare riferimento ad applicazioni in ambiente .NET sia web-oriented che desktop.
Il suo sito web è www.smelzo.it. Come sempre è a disposizione per ricevere suggerimenti o richieste sull'articolo all'indirizzo di posta elettronica**
francesco@smelzo.it

TRASFORMA I DATI IN GRAFICI

HAI REALIZZATO UN FANTASTICO SOFTWARE CHE ANALIZZA PUNTO PER PUNTO TUTTI I MOVIMENTI DELLA TUA AZIENDA. IL PROBLEMA È CHE IL TUO CAPO VUOLE VEDERE VISIVAMENTE L'ANDAMENTO DEI PRODOTTI, COME FARE? LA RISPOSTA È QUI…

Abbiamo realizzato un software per una società che vende prodotti on-line. Il programma conserva traccia degli acquisti dei clienti memorizzandoli in alcune tabelle di un database. Funziona alla perfezione ma il proprietario della società vorrebbe avere un quadro generale dell'andamento delle vendite, preferendo visualizzare dei grafici piuttosto che degli incomprensibili report pieni di numeri. Ci commissiona quindi un programma per Windows con qualche grafico a torta - per la distribuzione delle vendite per regione - e dei grafici a barre per l'andamento delle vendite nel corso dell'anno.

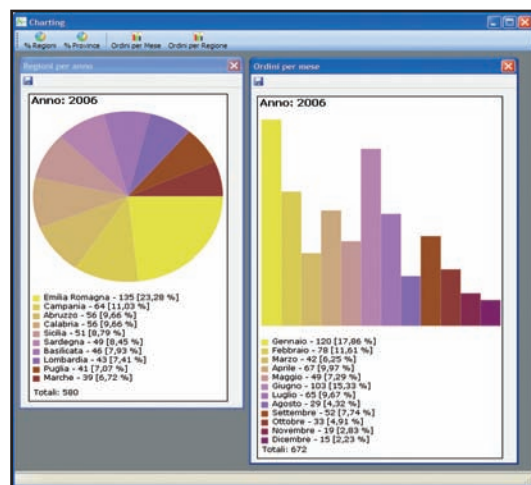



Fig. 1: Il progetto in esecuzione!

Analizziamo in poche righe la struttura del database. I prodotti sono suddivisi in categorie e sottocategorie; i clienti sono catalogati per regione e provincia. Nel momento in cui un cliente effettua un ordine, questo viene salvato nella tabella **OrdersHeader**: invece, la lista dei prodotti acquistati è salvata nella tabella **OrdersDetail**.
I dati sugli ordini vengono consolidati nella tabella **ProductsDestinations**, che contiene le informazioni su: anno, mese, regione, quantità di prodotti consegnati e volume occupato da tali prodotti.

Il consolidamento dei dati viene realizzato attraverso un pacchetto di SSIS chiamato *Products Destinations.dtsx* che si occupa di raggruppare i dati sulle vendite per mese e provincia. Il pacchetto è presente nel CD allegato, completo di script di installazione. La tabella **Products Destinations** è comunque già popolata con dei dati di test per consentire di procedere con gli esempi anche a chi non dovesse disporre di SSIS.

METTIAMOCI AL LAVORO!

Fortunatamente anche in questa occasione il framework .Net non ci abbandona al nostro destino… abbiamo infatti il namespace "**System.Drawing**", che fa proprio al caso nostro.
Utilizzeremo principalmente due classi:

- **Bitmap**
- **Graphics**

Possiamo pensare alla classe Bitmap come una tela su cui disegniamo; la classe Graphics potrebbe essere, invece, il nostro pennello.
Per creare una istanza della classe Bitmap basta definirne le dimensioni:

```
Dim Image As Bitmap = New Bitmap(CanvasWidth,
                                  CanvasHeight)
```

Una volta creata la tela, è possibile disegnarvi sopra utilizzando la classe Graphics:

```
Dim Graph As Graphics = Graphics.FromImage(Image)
```

Questa classe contiene una notevole varietà di metodi per disegnare oggetti come linee, cerchi o rettangoli. Possiamo suddividere i metodi per il disegno in due categorie:

- **Draw**Forma: come ad esempio DrawRectangle, sono i metodi per disegnare i contorni delle figure.

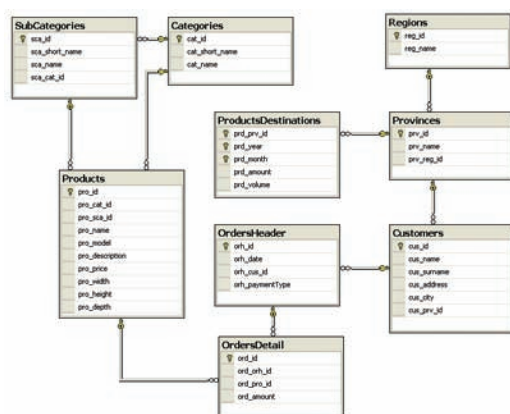



Fig. 2: Il database

- **Fill**Forma: come ad esempio FillRectangle, sono i metodi per colorare l'interno delle figure.

Le due viste sopra sono sicuramente le due classi "principe" del namespace, ma ce ne sono tante altre che utilizzeremo, spesso senza farci caso…
Ne cito alcune come esempio:

- *Brush*: il "pennello" con cui coloriamo gli oggetti
- *Font*: il carattere per il testo
- *Pen*: per disegnare le nostre linee.

Abbiamo poi alcune strutture come *Rectangle* che rappresenta un rettangolo o Point che rappresenta un punto.
Ma non c'è nulla di meglio per imparare ad utilizzare delle classi se non *utilizzandole*! Entriamo, quindi, nel vivo del nostro progetto e scriviamo le classi per realizzare i nostri grafici a barre e a torta.
Costruiremo due classi per il disegno di grafici a torta ed a barra. Entrambe le classi ereditano da una classe base **Chart**.

Creiamo un nuovo progetto di tipo "**Class Library**" con Visual Studio 2005 ed aggiungiamo quattro classi: *Chart.vb, PieChart.vb, BarsChart.vb e ChartingUtils.vb*. Le classi rappresenteranno un layer "puramente" grafico; si occuperanno esclusivamente di disegnare forme e non, ad esempio, del recupero dei dati, che verranno reperiti esternamente ed inviati alle classi tramite una DataTable.
Le proprietà della classe sono quelle comuni a tutti i grafici che andremo a costruire.

- **_Data**: la DataTable con i dati per il grafico
- **_Title**: il titolo del grafico
- **_ValuesColumn**: la DataColumn contenente i valori
- **_DescriptionsColumn**: la DataColumn contenente i dati
- **_LegendFont**: il carattere per la legenda
- **_TitleFont**: il carattere per il titolo

Abbiamo poi dei metodi di disegno:

- **GetColors**: recupera una lista di colori per il grafico
- **DrawTitle**: disegna il titolo del grafico
- **DrawLegend**: disegna la legenda
- **DrawTotals**: scrive la riga con i totali

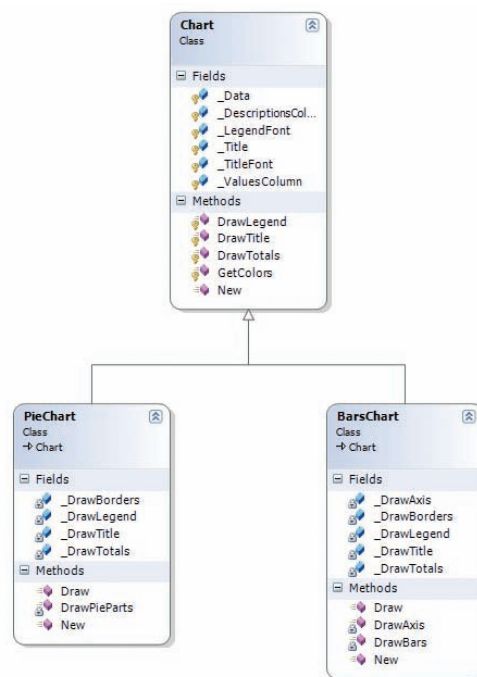



Fig. 3: Le classi del nostro progetto

La funzione **DrawTitle** crea, come prima cosa, la formattazione per il testo e l'allineamento:

```
Dim Format As StringFormat = New StringFormat()
Format.Alignment = StringAlignment.Near
Format.LineAlignment = StringAlignment.Near
```

Quindi scrive il titolo utilizzando il metodo **DrawString**:

```
Graph.DrawString( _
       _Title, _TitleFont, Brush, _
       New Rectangle(0, 0, Height, TitleHeight), Format)
```

Il metodo prevede come parametro un oggetto di tipo **Rectangle**, che rappresenta i limiti del testo.
Il metodo **DrawLegend** è più complicato. Cicla sulle righe della *DataTable* e per ogni elemento trovato: disegna un rettangolo del colore giusto di dimensioni 10x10,

```
Graph.FillRectangle(CType(Colors(i), SolidBrush), 5, _
    GraphHeight - LegendHeight + _LegendFont.Height *
                                          i + 5, 10, 10)
```

IL CODICE ALLEGATO

Gli esempi nell'articolo sono in Visual Basic; per chi fosse interessato, nel CD allegato è presente tutto il codice sorgente anche in C#.

scrive la descrizione associata alla riga della *DataTable*.

```
Graph.DrawString( _
    _Data.Rows(i)(_DescriptionsColumn) & " - " & _
    _Data.Rows(i)(_ValuesColumn), _
    _LegendFont, Brush, _
    20, GraphHeight - LegendHeight +
                        _LegendFont.Height * i + 1)
```

Potete trovare la classe nel CD allegato.
Analizziamo adesso la classe per il disegno dei grafici a torta.
Il cuore della classe è la funzione **Draw** che si occupa di disegnare il grafico e restituire un oggetto di tipo **Bitmap**.
Nelle prime righe, vengono calcolate larghezza ed altezza del grafico in funzione dei parametri passati.
In particolare, l'altezza delle legenda è data dall'altezza del font per la legenda moltiplicato per il numero di elementi più la spaziatura.

```
LegendHeight = _LegendFont.Height *
                (_Data.Rows.Count) + SpacingHeight
```

Mentre l'altezza totale è data da:

```
GraphHeight = PieHeight + LegendHeight + TitleHeight
                                    + SpacingHeight
```

ossia l'altezza del grafico più quella della legenda, del titolo e della spaziatura. Impostate le dimensioni del grafico, carichiamo la lista dei colori che serviranno; quindi i totali, sommando i valori delle *DataRow* appartenenti alla *DataTable*.
Impostati i parametri iniziali, possiamo creare la base del grafico:

```
' Creo l'immagine
Dim Image As Bitmap = New Bitmap(GraphWidth,
            GraphHeight , PixelFormat.Format64bppArgb)
Dim Graph As Graphics = Graphics.FromImage(Image)
Graph.SmoothingMode = SmoothingMode.HighQuality
Graph.TextRenderingHint =
                    TextRenderingHint.AntiAliasGridFit

' contenitore della torta
Dim PieRectangle As Rectangle = New Rectangle(5,
        TitleHeight, GraphWidth - 10, PieHeight - 10)
' sfondo del grafico
Graph.FillRectangle(New SolidBrush(Color.White), 0, 0,
                            GraphWidth, GraphHeight)
```

Creiamo un oggetto **Bitmap** (la nostra tela) impostando larghezza ed altezza dell'immagine, e un parametro opzionale "*PixelFormat*" che indica la profondità di colore.

Creiamo quindi l'oggetto **Graph** utilizzando il metodo **GraphicsFromImage**; verrà utilizzato per disegnare i nostri elementi nella Bitmap.
Impostiamo quindi due parametri *SmoothingMode* e *TextRenderingHint* per migliorare la qualità del grafico. Definiamo infine un **Rectangle**, che sarà il contenitore della torta e coloriamolo di bianco.
A questo punto siamo pronti per disegnare gli elementi del grafico!
Partiamo dal titolo con il metodo *DrawTitle* della classe base, disegniamo quindi le fette della torta ed infine la legenda e i totali rispettivamente con i metodi *DrawLegend* e *DrawTotals* della classe base.
Analizziamo in dettaglio il metodo per disegnare le fette della torta:

```
' angolo corrente
Dim CurrentAngle As Single = 0.0F
' fette della torta
For i As Integer = 0 To _Data.Rows.Count - 1
        DrawPieParts(Colors, Totals, CurrentAngle, Graph,
                                        PieRectangle, i)
Next
```

Impostiamo a zero il valore dell'angolo corrente; cicliamo quindi le righe della *DataTable* per disegnare le fette della torta tramite il metodo **DrawPieParts**.
La funzione DrawPieParts

```
Dim DrawAngle As Single = _
    Convert.ToSingle(_Data.Rows(i)(_ValuesColumn)) /
                                        Totals * 360
If (_DrawBorders) Then
    Graph.DrawPie(New Pen(Brushes.Black),
                PieRectangle, CurrentAngle, DrawAngle)
End If
Graph.FillPie(CType(Colors(i), SolidBrush), PieRectangle,
                            CurrentAngle, DrawAngle)

' incremento il valore dell'angolo corrente
CurrentAngle += DrawAngle
```

utilizza i metodi **DrawPie** e **FillPie** per disegnare i contorni e riempire le parti del grafico.
Viene, quindi, incrementato il valore dell'angolo corrente della quantità appena disegnata.
Il valore dell'angolo da disegnare viene calcolato come percentuale del valore della riga corrente sul totale e poi moltiplicato per 360 per ottenere il valore in gradi.
Occupiamoci adesso della classe **BarsChart** per il disegno dei grafici a barre di cui analizzeremo soltanto i punti salienti.

```
[...]

Public Class BarsChart
```

```
Inherits Chart
 [...]
Private _DrawAxis As Boolean
Public Sub New( _
    ByVal Data As DataTable, ByVal Title As String, _
    ByVal ValuesColumn As String, ByVal
                      DescriptionsColumn As String, _
    ByVal LegendFont As Font, ByVal TitleFont As Font,
                                                 _
    ByVal DrawTitle As Boolean, ByVal DrawBorders As
                                           Boolean, _
    ByVal DrawAxis As Boolean, ByVal DrawLegend As
                                           Boolean, _
    ByVal DrawTotals As Boolean)
    MyBase.New( _
        Data, Title, _
        ValuesColumn, DescriptionsColumn, _
        LegendFont, TitleFont)
     [...]
    _DrawAxis = DrawAxis
End Sub
Public Function Draw( _
    ByVal Width As Integer, ByRef GraphWidth As
                                           Integer, _
    ByRef GraphHeight As Integer) As Bitmap
    ' Imposto larghezza ed altezza immagine
    [...]
    ' Imposto altezza e larghezza del grafico
    [...]
    ' Le barre
    Dim BarHeight As Single = 0
    Dim BarOrigin As PointF = New PointF
                                      (GraphLeft, 0)
    For i As Integer = 0 To _Data.Rows.Count - 1
        DrawBars(BarsHeight, GraphTop, BarsWidth,
                                           Colors, _
             MaxValue, Graph, BarHeight, BarOrigin, I,
                                              Brush)
    Next
    ' Disegno gli assi
    If (_DrawAxis) Then
        DrawAxis(GraphWidth, BarsHeight, GraphTop,
                                       GraphLeft, Graph)
    End If
    [...]
    Graph.Dispose()
    Return Image
End Function
Private Sub DrawAxis( _
    ByVal GraphWidth As Integer, ByVal BarsHeight As
                                           Integer, _
    ByVal GraphTop As Integer, ByVal GraphLeft As
                           Integer, ByVal Graph As Graphics)
    Graph.DrawLine( _
        New Pen(Color.Black, 2), New Point(GraphLeft,
                                       GraphTop - 10), _
        New Point(GraphLeft, GraphTop + BarsHeight))
    Graph.DrawLine( _
        New Pen(Color.Black, 2), New Point(GraphLeft,
                              GraphTop + BarsHeight), _
        New Point(GraphLeft + GraphWidth, GraphTop
                                       + BarsHeight))
End Sub
Private Sub DrawBars( _
    ByVal BarsHeight As Integer, ByVal GraphTop As
                   Integer, ByVal BarsWidth As Integer, _
    ByVal Colors As ArrayList, ByVal MaxValue As
                        Single, ByVal Graph As Graphics, _
    ByRef BarHeight As Single, ByRef BarOrigin As
                              PointF, ByVal i As Integer, _
    ByVal Brush As SolidBrush)
    BarHeight =
    Convert.ToSingle(_Data.Rows(i)(_ValuesColumn)) *
                                BarsHeight / MaxValue
    BarOrigin.Y = GraphTop + BarsHeight - BarHeight

    'disegno barra (ed eventualmente contorni)
    If (_DrawBorders) Then
        Graph.DrawRectangle( _
            New Pen(Brush), BarOrigin.X, BarOrigin.Y, _
            BarsWidth, BarHeight)
    End If

    Graph.FillRectangle( _
            CType(Colors(i), SolidBrush), BarOrigin.X, _
            BarOrigin.Y, BarsWidth, BarHeight)

    'disegno il valore in alto
    Graph.DrawString( _
             (_Data.Rows(i)(_ValuesColumn)).ToString(),
                                                    _
            _LegendFont, Brush, BarOrigin.X + 4,
                                     BarOrigin.Y - 16)

    'reimposto l'origine della barra
    BarOrigin.X = BarOrigin.X + BarsWidth

 End Sub
End Class
```

Le dimensioni vengono calcolate in modo molto simile a quelle del grafico a torta; quello che cambia in modo sostanziale è il contenuto.
Per il disegno cicliamo sulle *DataRow* e per ciascuna riga eseguiamo il metodo **DrawBars** che si occupa di disegnare la singola barra.
Per farlo utilizziamo un oggetto di tipo *PointF* che chiamiamo “*BarOrigin*” e che rappresenta il punto di origine della barra. L’origine viene impostata di volta in volta a partire dalla dimensione che dovrà avere la barra. L’altezza sarà in proporzione al valore massimo in modo da avere dei grafici che restano all’interno della superficie di disegno.

```
BarHeight =
    Convert.ToSingle(_Data.Rows(i)(_ValuesColumn)) *
```

SYSTEM. DRAWING

Il namespace System.Drawing fornisce una serie di classi per creare o modificare immagini di diverso formato. Utilizzando queste classi è possibile creare grafici con dati recuperati da database, oppure modificare immagini preesistenti. Ulteriori informazioni e la lista completa di classi e metodi sono disponibili all’indirizzo: http://msdn2.microsoft.com/it-it/library/system.drawing(VS.80).aspx

```
                                    BarsHeight / MaxValue
BarOrigin.Y = GraphTop + BarsHeight - BarHeight
```

La coordinata Y del punto viene calcolata come abbiamo visto perché il sistema di assi cartesiani ha come origine il punto, in alto a sinistra. La figura in basso può aiutarci a capire meglio.

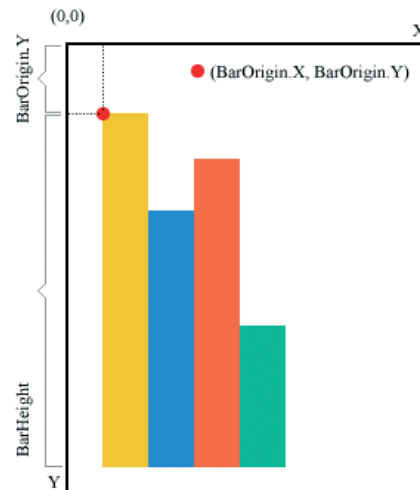



Fig. 3: Calcoliamo il vertice della barra

Ad ogni ciclo, modifichiamo la posizione della barra successiva in modo che le barre vicine non si sovrappongano.

```
BarOrigin.X = BarOrigin.X + BarsWidth
```

Per il disegno utilizziamo i metodi *DrawRectangle* per i bordi, *FillRectangle* per il riempimento e *DrawString* per scrivere il valore corrispondente alla singola barra.
La classe **Utils** è molto semplice: contiene, infatti, un solo metodo statico per recuperare un colore a partire da un numero intero e viene usata per colorare dinamicamente gli elementi dei grafici.

IL PROGETTO

Scritto il codice delle classi per il disegno, prepariamo le form per la visualizzazione dei grafici. A tale scopo creiamo un progetto Windows Forms chiamato Charting.
Aggiungiamo, come prima cosa, il riferimento al progetto ChartingLibrary per poter disporre delle classi appena trovate.
Il progetto è composto dai seguenti oggetti:

- una form Main che contiene i pulsanti per avviare i nostri report
- alcune form in cui inserire i parametri e lanciare i report (MonthAmount, MonthAmountByRegion, ProvincesByYear e RegionsByYear)
- due form per la visualizzazione del grafico: BarsGraph e PieGraph)
- una classe utils per i metodi di utilità

COSTRUIAMO LE FORM PER I GRAFICI

Analizziamo la costruzione delle form BarsGraph e PieGraph che ci consentiranno di visualizzare i grafici.
Creiamo una nuova form chiamata *BarsGraph*.
Inseriamo nella form una *ToolStrip* e, al suo interno, un pulsante "*btn_save*"; quindi un *SaveFileDialog* "*dia_save*" per gestire il salvataggio dell'immagine.
Al centro della form, aggiungiamo una *PictureBox* che chiamiamo "pbx_imageContainer".
Analizziamone il codice: abbiamo delle proprietà ed il costruttore "overloaded" della form per valorizzarle.

```
' la datatable con i dati
Private _Data As DataTable
' la query
Private _Sql As String
' titolo della form
Private _FormTitle As String
' titolo del grafico
Private _GraphTitle As String
' se disegnare il titolo
Private _DrawTitle As Boolean
' se disegnare i bordi
Private _DrawBorders As Boolean
' se disegnare la legenda
Private _DrawLegend As Boolean
' se visualizzare i totali
Private _DrawTotals As Boolean
' larghezza del grafico
Private _GraphWidth As Integer
Public Sub New( _
    ByVal FormTitle As String, ByVal GraphTitle As String,
                                                        _
    ByVal GraphWidth As String, ByVal Sql As String, _
    ByVal DrawTitle As Boolean, ByVal DrawBorders As
                                             Boolean, _
    ByVal DrawLegend As Boolean, ByVal DrawTotals As
                                              Boolean)
    InitializeComponent()
    _Sql = Sql
    _GraphTitle = GraphTitle
    _FormTitle = FormTitle
    _DrawTitle = DrawTitle
    _DrawBorders = DrawBorders
    _DrawLegend = DrawLegend
    _DrawTotals = DrawTotals
    _GraphWidth = GraphWidth
End Sub
```

Nell'evento "Load" della form carichiamo i dati in base ai parametri ricevuti.

```
Try
    Me.Text = _FormTitle
    Dim ConnectionString As String = _
ConfigurationManager.ConnectionStrings
                    ("connectionString").ConnectionString
    Dim Connection As SqlConnection = New
                        SqlConnection(ConnectionString)
    Connection.Open()
    Try
        Dim Command As SqlCommand = New
                        SqlCommand(_Sql, Connection)
        Dim Data As DataSet = New DataSet()
        Dim DataAdapter As SqlDataAdapter = New
                            SqlDataAdapter(Command)
        DataAdapter.Fill(Data)
        _Data = Data.Tables(0)
    Finally
        Connection.Close()
    End Try
    If (_Data.Rows.Count > 0) Then
        ' Disegno il grafico
        DrawGraph()
    Else
        MessageBox.Show("Nessun dato per il periodo
                                    selezionato")
    End If
Catch ex As Exception
    MessageBox.Show("Errore: " + ex.Message)
End Try
```

Assegniamo il titolo alla form, quindi riempiamo la *DataTable _Data* a partire dalla *query _Sql*. Costruita la *DataTable*, lanciamo la funzione *DrawGraph*.

```
Private Sub DrawGraph()
    Try
        ' definisco i font per titolo e legenda
        Dim LegendFont As Font = New Font("Verdana", 9)
        Dim TitleFont As Font = New Font("Verdana", 12,
                                        FontStyle.Bold)
        ' istanzio la classe
        Dim bc As BarsChart = New BarsChart( _
            _Dati, _GraphTitle, _
            "VAL", "DES", LegendFont, TitleFont, _
            _DrawTitle, _DrawBorders, _DrawLegend,
                            _DrawTotals, _DrawAxis)
        ' dimensioni iniziali grafico
        Dim GraphHeight As Integer =
                        pbx_imageContainer.Height
        Dim GraphWidth As Integer = _GraphWidth
        pbx_imageContainer.Width = _GraphWidth - 30
        ' creo il grafico
        Dim I As Image = _
            CType(bc.Draw(pbx_imageContainer.Width - 4,
```

```
                        GraphWidth, GraphHeight), Image)
        ' reimposto le dimensioni della picture e della form
        pbx_imageContainer.Width = GraphWidth + 4
        Me.Width = GraphWidth + 40
        pbx_imageContainer.Height = GraphHeight + 4
        Me.Height = GraphHeight + 80
        ' Assegno l'immagine
        pbx_imageContainer.Image = I
    Catch ex As Exception
        MessageBox.Show("Errore: " + ex.Message)
    End Try
End Sub
```

Non facciamo altro se non istanziare la classe *BarsChart* e richiamare il metodo *Draw* per ottenere un oggetto di tipo *Image*.
Associamo quindi l'oggetto *all'ImageContainer* ed infine reimpostiamo le dimensioni della form per poter visualizzare l'intero grafico. Nel CD allegato trovate anche un esempio associato al metodo STN_SAVE che consente di salvare il grafico creato su File.

VEDIAMO UN ESEMPIO

A questo punto non ci resta che preparare le query da inviare alle form di disegno. Iniziamo con una form che ci consente di visualizzare la quantità di prodotti ordinati in un anno, raggruppati per mese.
La query base è la seguente:

```
SELECT
  sum(prd_amount) VAL,
  mon_name DES
FROM ProductsDestinations
INNER JOIN Months
   ON prd_month = mon_id
WHERE prd_year = 2006
GROUP BY prd_month, mon_name
ORDER BY prd_month
```

Procediamo con la costruzione della form che chiamiamo *MonthAmount*.
Inseriamo:

- un *NumericUpDown txt_year* per l'anno
- un *NumericUpDown txt_graphWidth* per la larghezza del grafico
- delle *CheckBox* per le opzioni

Il codice del pulsante Ok è molto semplice:

```
Try
    Dim Sql As String = _
        " SELECT " & _
        "  sum(prd_amount) VAL," & _
        "  mon_name DES" & _
        " FROM ProductsDestinations" & _
```

```
          " INNER JOIN Months" & _
          "   ON prd_month = mon_id" & _
          " WHERE " & _
          "   prd_year = " & txt_year.Value & _
          " GROUP BY prd_month, mon_name" & _
          " ORDER BY prd_month"
    Dim bg As BarsGraph = New BarsGraph( _
          Me.Text, "Anno: " & txt_year.Value,
                Convert.ToInt32(txt_graphWidth.Value), _
          Sql, _
          chk_DrawTitle.Checked,
                        chk_DrawBorders.Checked, _
          chk_DrawLegend.Checked,
           chk_DrawTotals.Checked, chk_DrawAxis.Checked)
    bg.MdiParent = Me.MdiParent
```

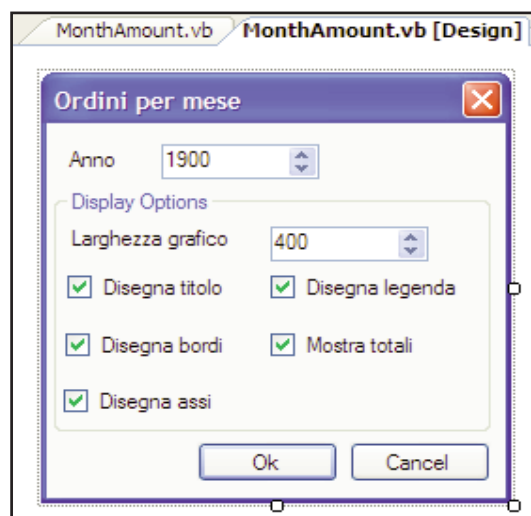



Fig. 5: La form con i parametri

```
    bg.Show()
    Me.Close()
Catch ex As Exception
    MessageBox.Show("Errore: " + ex.Message)
End Try
```

Non fa altro che istanziare BarsGraph con i parametri che abbiamo definito e mostrare la form.
A questo punto abbiamo tutti gli strumenti per sbizzarrirci nella creazione di grafici a torta e a barre; nel CD allegato ci sono altri tre esempi con query e grafici diversi.

E IL WEB?

Abbiamo presentato il progetto al committente che è rimasto entusiasta. Qualche giorno dopo però ci telefona e ci dice: i grafici erano davvero belli, ma se volessi vederli su web?
Niente paura! Le classi che abbiamo creato possono essere utilizzate anche per generare delle immagini al volo su un sito web. Vediamo come fare grazie ad un semplice esempio:

Creiamo un progetto Web e referenziamo la libreria *ChartingLibrary*, quindi una *WebForm* chiamata *RegionsByYearGraph*.
Ecco il codice della pagina:

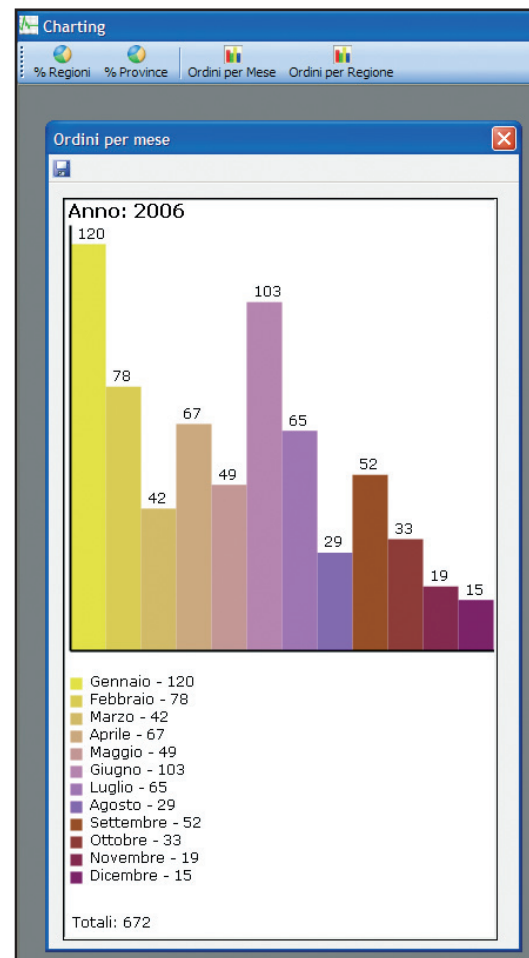



Fig. 6: Il nostro grafico in azione!

```
Imports System.Data
Imports System.Drawing
Imports System.Drawing.Imaging
Imports System.Data.SqlClient
Imports System.IO
Imports ChartingLibrary
Partial Class ImagePieChart
    Inherits System.Web.UI.Page
    Private _Data As DataTable
    Protected Sub Page_Load(ByVal sender As Object,
            ByVal e As System.EventArgs) Handles Me.Load
        Dim ImageWidth As Integer = 200
        If Not Request.QueryString("Width") Is Nothing
                                                  Then
            ImageWidth =
             Convert.ToInt32(Request.QueryString("Width"))
        End If
        Dim Year As Integer = DateTime.Now.Year
        If Not Request.QueryString("Year") Is Nothing Then
            Year =
```

```
        Convert.ToInt32(Request.QueryString("Year"))
    End If
    Dim ConnectionString As String =
            ConfigurationManager.ConnectionStrings
            ("connectionString").ConnectionString
    Dim Connection As SqlConnection = New
            SqlConnection(ConnectionString)
    Connection.Open()
    Try
        Dim Sql As String = _
            " SELECT " & _
              "  sum(prd_amount) VAL," & _
              "  reg_name DES" & _
              " FROM ProductsDestinations" & _
              " INNER JOIN Provinces" & _
              "   ON prd_prv_id = prv_id" & _
              " INNER JOIN Regions" & _
              "   ON prv_reg_id = reg_id" & _
              " WHERE " & _
              "   prd_year = " & Year.ToString() & _
              " GROUP BY reg_name" & _
              " ORDER BY sum(prd_amount) DESC"
        Dim Command As SqlCommand = New
                SqlCommand(sql, Connection)
        Dim Data As DataSet = New DataSet()
        Dim DataAdapter As SqlDataAdapter = New
                SqlDataAdapter(Command)

        DataAdapter.Fill(Data)
        _Data = Data.Tables(0)
        ' Disegno il grafico
        DrawGraph(ImageWidth)
    Finally
        Connection.Close()
    End Try
  End Sub
  Private Sub DrawGraph(ByVal ImageWidth As Integer)
    ' definisco i font per titolo e legenda
    Dim LegendFont As Font = New Font("Verdana", 9)
    Dim TitleFont As Font = New Font("Verdana", 12,
            FontStyle.Bold)
    Dim pc As PieChart = New PieChart( _
        _Data, "Regioni per anno", _
        "VAL", "DES", LegendFont, TitleFont, _
        True, True, True, True)
    Dim I As Image = CType(pc.Draw(ImageWidth -
            4, ImageWidth), Image)
    Dim target As Stream = Response.OutputStream

    I.Save(target, ImageFormat.Jpeg)
  End Sub
End Class
```

Leggiamo due parametri:

- l'anno: *Request.Query String("Year")*
- la dimensione del grafico: *Request.QueryString ("Width")*

Impostiamo la query e riempiamo la *DataTable*. Fin qui, tutto come per la nostra applicazione *WindowsForm…*
È la funzione *DrawGraph* ad essere leggermente diversa, anche se solo nella parte finale.

```
' creo il grafico
Dim I As Image = CType(pc.Draw(ImageWidth - 4,
        ImageWidth), Image)
Dim target As Stream = Response.OutputStream
I.Save(target, ImageFormat.Jpeg)
```

Infatti, una volta ottenuto l'oggetto *Image*, non lo associamo ad un controllo ma lo inviamo direttamente all'*OutputStream* mediante il metodo *Save*.
Abbiamo la pagina che crea l'immagine, non ci resta che utilizzarla.
Farlo è molto semplice: è sufficiente creare una seconda *WebForm* contenente due *TextBox* per i parametri ed un Tag *asp:Image*.

```
<asp:Image ID="img_graph" runat="server"
        ImageUrl="" Visible="false"/>
```

Aggiungiamo, quindi, un *Button* con il seguente codice associato all'evento *Click*:

```
If (txt_width.Text <> "") Then
    img_graph.ImageUrl = _
        "~/RegionsByYearGraph.aspx?Year=" & _
        txt_anno.Text & "&Width=" & txt_width.Text
    img_graph.Visible = True

End If
```

Così facendo, quando clickiamo sul pulsante, l'immagine generata al volo dalla pagina *RegionsByYear Graph.aspx* viene associata al controllo *img_graph*.
Non ci resta che compilare ed eseguire!

Carmelo Scuderi

NOTE SULL'INSTALLAZIONE DEL PACKAGE DI SSIS

**Nella cartella "2. SSIS Package" è presente il file "deploy.bat". Apriamolo e cambiamo il nome del database per farlo puntare al nostro server. Eseguiamolo.
A questo punto il package dovrebbe essere memorizzato sul nostro database.**

Una volta installato il package siamo pronti per eseguirlo. Possiamo farlo tramite lo script T-SQL "ExecPackege.sql" presente nella stessa cartella del package.

Eseguendo il comando xp_cmdshell, potremmo ricevere un errore del genere:

**Msg 15281, Level 16, State 1, Procedure xp_cmdshell, Line 1
SQL Server blocked access to procedure 'sys.xp_cmdshell' […]**

Nel caso dovesse succedere, occorre impostare un parametro nella "Surface Area Configuration for Features" per consentire l'esecuzione della stored xp_cmdshell

SECOND LIFE REZ & DETECT

IL MONDO DI SL NON E' STATICO MA È FATTO DI AVATAR IN MOVIMENTO CHE INTERAGISCONO FRA LORO. COME ACCORGERSI DELLA PRESENZA DI UN AVATAR O DI UN OGGETTO NELL'AREA DI INFLUENZA DI UN ALTRO? ECCO I METODI DI LINDEN...

Dopo aver affrontato, nel precedente articolo, alcuni concetti di base del Linden Scripting Language, proseguiamo nell'analisi del linguaggio, cercando di affrontare argomenti più avanzati.
Chiaramente, essendo questa una sorta di "seconda puntata" della saga dedicata a Second Life, considereremo noti tutti i concetti espressi nel primo articolo dedicato al nuovo mondo; in particolare la modalità di registrazione ed accesso a Second Life, le basi per creare un oggetto e le modalità per inserire nell'oggetto, del codice di scripting.
Questo presupposto ci consente di andare più spediti nella trattazione dei concetti alla base del presente articolo, e, soprattutto, ci consente di evitare di ripetere i concetti di base.
In questa seconda puntata analizzeremo gli aspetti "reconditi" legati alla creazione degli oggetti e cominceremo ad affrontare l'argomento della "detection" degli avatar, ovvero come sia possibile individuare gli avatar che, in qualche modo, vengono in contatto con i nostri oggetti.

REZ IT!

Un aspetto interessante da affrontare è la fase di creazione di un oggetto; in gergo la fase di *rez*, ovvero la fase di creazione di un oggetto.
Innanzitutto dobbiamo sapere che ogni oggetto creato in Second Life è individuato univocamente dalla sua chiave che è un identificatore univoco, o UUID (Universal Unique IDentifier), rappresentato da una stringa di numeri esadecimali di 36 caratteri nel formato "00000000-0000-0000-0000-000000000000" (per esempio "66864f3c-e095-d9c8-058d-d6575e6ed1b8"). Questa chiave sarà la via per comunicare con il nostro oggetto, dato che tutte le funzioni di libreria, utili per la comunicazione, prendono come primo parametro proprio la chiave del destinatario.
Normalmente, una volta che abbiamo sviluppato il nostro oggetto, sia in termini grafici che di funzionalità interattive, lo riponiamo nel nostro inventario per poterlo riutilizzare in seguito.
Questa operazione, nasconde una insidia che dobbiamo analizzare per evitare malfunzionamenti dei nostri script che si basano sulle chiavi degli oggetti.
In effetti l'inventario del nostro avatar NON è una semplice "borsa" in cui poniamo gli oggetti. In verità, quando poniamo un oggetto nell'inventario (clic con il tasto destro del mouse sull'oggetto, quindi clic sulla voce "Take") noi cancelliamo l'oggetto dal terreno e portiamo nell'inventario un suo modello; ogniqualvolta estraiamo nuovamente l'oggetto dall'inventario, ne creiamo una nuova copia. Per dirla in un altro modo (comprensibile a chi programma ad oggetti), è come se ponessimo nell'inventario una classe, ed ogni volta che estraiamo l'oggetto ne creiamo una sua istanza. Queste precisazioni sono molto importanti dato che la chiave dell'oggetto cambia ogni volta (dato che in verità viene creato un nuovo oggetto), quindi se creiamo script basati sulla chiavi dobbiamo stare molto attenti ai processi di *rez* e *de-rez*. Come al solito, dato che anche in questo "episodio" utilizzeremo l'approccio "hands-on-lab", rivediamo tutto ciò che abbiamo detto tramite un esempio pratico.
Innanzitutto creiamo un oggetto, andiamo nella parte dedicata agli script ed inseriamo il

ETIMOLOGIA DEL REZ

Lo strano "slang" usato per indicare la creazione di un oggetto, in realtà è un termine coniato negli anni '80 quando, nel film Tron, la parola "de-rezz" veniva utilizzata per indicare che un oggetto del mondo 3D veniva distrutto o "de-resolved". La parola usata al contrario (quindi senza il prefisso "de") indica, quindi, la creazione di un oggetto.

seguente codice (contenuto nel file *rezIt.lsl* a supporto dell'articolo):

```
default
{

 // Scatenato dal tocco di un avatar
 touch_start(integer total_number)
 {

  // Invia un messaggio istantaneo al proprietario
  //dell'oggetto
  llOwnerSay((string)llGetKey());
 }

 // Scatenato ogni volta che l'oggetto
//  viene creato dall'inventario
 on_rez(integer start_param)
 {
  // Invia un messaggio istantaneo
  //al proprietario  dell'oggetto

  llOwnerSay((string)llGetKey());

  // Resetta lo script
  llResetScript();
 }
}
```

Incontriamo per la prima volta l'evento *on_rez*, che viene scatenato ogniqualvolta un oggetto viene creato a partire dall'inventario. All'interno dell'evento, utilizziamo la funzione di libreria *llOwnerSay* che invia un messaggio al proprietario dell'oggetto (la stessa funzione la usiamo anche per gestire il tocco dell'oggetto da parte di un avatar). Il messaggio è proprio la chiave dell'oggetto che ricaviamo invocando la funzione di libreria *llGetKey*. È appena il caso di notare come il risultato dell'invocazione della funzione di libreria *llGetKey* (che è un valore di tipo *key*), venga "castato" al tipo stringa in modo da evitare un errore di tipo (dato che la funzione di libreria *llOwnerSay* accetta come parametro valori di tipo stringa).
La seconda funzione di libreria invocata nell'evento, *llResetScript*, ci consente di aprire una piccola parentesi, che riguarda un ulteriore aspetto della programmazione in Linden Scripting Language fonte di numerosi malfunzionamenti. Facendo un piccolo passo indietro, ci ricordiamo che ogni script rappresenta la programmazione di una macchina a stati finiti orientata agli eventi. Il nostro script viene lanciato la prima volta che viene salvato dall'editor, quindi evolve nel susseguirsi degli stati che rispondono agli eventi scatenati dall'oggetto o nell'oggetto. Ebbene una cosa che spesso si dimentica è che quando portiamo l'oggetto nell'inventario, questo viene memorizzato nel suo stato attuale; ciò significa che quando lo ricreiamo a partire dall'inventario, questo NON "riparte" dallo stato di *default* ma dallo stato in cui lo avevamo lasciato quando lo avevamo messo nell'inventario. Di conseguenza anche tutte le variabili globali NON hanno il valore iniziale ma quello che avevano al momento dell'inserimento nell'inventario. In realtà, quando creiamo un nuovo oggetto a partire dall'inventario, la nostra intenzione è quella di creare un oggetto "nuovo" (appunto), quindi che parta dallo stato di default con tutte le variabili con i valori predefiniti. Ebbene la funzione di libreria *llResetScript* riporta esattamente lo script allo stato iniziale, in modo da avere un oggetto "nuovo". Ovviamente dovremo inserire il codice relativo all'evento *on_rez* in TUTTI gli stati che implementeremo per il nostro oggetto.
Chiusa la parentesi, vediamo come funziona il nostro oggetto nella sequenza mostrata in figura 1.

Figura 1: La variazione della chiave dell'oggetto.

Se proviamo a descrivere le tre scene della sequenza:

1) Creiamo l'oggetto, inseriamo il codice precedete, lo salviamo quindi lo tocchiamo: la sua chiave viene ci viene visualizzata

UN SACCO DI OBJECT

Un piccolo suggerimento sulla denominazione degli oggetti: ogni volta che creiamo un nuovo oggetto, il simulatore lo nomina con l'anonimo "Object". Ebbene è utile andare a modificare subito questo parametro in modo che, quando lo porteremo nel nostro inventario, avremo modo di riconoscerlo immediatamente.
Nell'inventario, infatti, gli oggetti sono elencati mostrando il loro "nome"; se non adottiamo un protocollo di denominazione degli oggetti, avremo presto un sacco di "Object" che non sapremo riconoscere.

NOTA

COMUNICAZIONI

Una interessante sezione del wiki sul Linden Scripting Language, tratta le diverse funzioni di libreria da utilizzare per inviare comunicazioni di vario genere: http://lslwiki.net/lslwiki/wakka.php?wakka=communications.

attraverso un messaggio istantaneo.
2) Poniamo l'oggetto nell'inventario (clic con il tasto destro del mouse sull'oggetto, quindi click sulla voce "Take"), dopo avergli cambiato nome per riconoscerlo più facilmente nell'inventario (ad esempio lo chiamiamo *on_rez*).
3) Creiamo nuovamente l'oggetto dall'inventario; viene invocato il metodo *on_rez* che ci mostra la chiave del nuovo oggetto e verifichiamo che, in effetti, la chiave è cambiata.

Al fine di evitare malfunzionamenti dovuti a errate impostazioni delle chiavi degli oggetti, è bene tenere traccia delle suddette chiavi con il semplice codice appena visto che, tra l'altro, è assolutamente trasparente al "mondo" dato che colloquia con il solo proprietario dell'oggetto inviandogli messaggi istantanei privati.

EDITOR ALTERNATIVI

Sebbene il client di SL consenta da solo di sviluppare codice, è interessante dare uno sguardo alla pagina http://lslwiki.net/lslwiki/wakka.php?wakka=AlternativeEditors dove sono elencati tutta una serie di editor installabili sulla nostra macchina ed utili per continuare a sviluppare anche in assenza di connessione al "mondo". Tra questi è da segnalare LSL-Editor che, tra l'altro, fornisce un "intellisense" tipo Visual Studio, colorazione del codice, e un debugger(http://www.lsleditor.org/). Sebbene il programma non necessiti di installazione (basta scaricarlo, unzipparlo e lanciarlo), gira solo in ambiente Windows con .Net Framework versione 2.0, installato.

DETECT IT!

Il secondo aspetto "avanzato" che vogliamo affrontare riguarda l'argomento della "detection" degli avatar. C'è un aspetto fondamentale in Second Life che è il caso di mettere in evidenza. In Second Life ogni "residente" (o avatar che dir si voglia) è ben identificato all'interno del "mondo". Questo vuol dire che, a differenza di ciò che accade per i siti web, noi abbiamo la possibilità di sapere esattamente chi visita i nostri negozi o i nostri luoghi. In altre parole ciò vuol dire che in Second Life non esiste l'anonimato (da non confondere con l'incognito; si veda la nota a lato). Possiamo immaginare da soli che potenziale ci offre questa caratteristica di Second Life; a differenza dei siti web in cui possiamo al più avere una collezione di "hit" ovvero una serie di indirizzi IP che fanno riferimento alle pagine visitate, in Second Life possiamo potenzialmente identificare nome e cognome (di Second Life ovviamente) di ogni visitatore.
Ci sono diverse tecniche che ci consentono di individuare gli avatar che ci visitano; possiamo mettere in relazione le tecniche con gli eventi che ci consentono l'individuazione ovvero gli eventi scatenati dal *tocco*, dai *sensori* e dalle *collisioni* tra gli avatar ed il nostro oggetto.
Le tecniche differiscono fondamentalmente dal tipo di interazione che deve effettuare l'avatar per essere individuato; in ogni caso utilizzeremo un insieme di funzioni di libreria tutte individuate dal prefisso: *llDetected* (indicheremo le suddette funzioni anche con la sigla *llDetected**).

INCOGNITO VS. ANONIMATO

Il fatto che in Second Life non esista anonimato è conseguenza della creazione dell'avatar stesso che, per sua natura, ha un nome, un cognome e (soprattutto) una chiave che lo identifica univocamente all'interno di Second Life. Ovviamente ciò NON vuol dire che se ci registriamo in Second Life ogni sconosciuto ha la possibilità di venire a bussare alla nostra porta (reale). In fase di registrazione è ancora possibile (anche se non consigliabile) fornire le generalità del cittadino più famoso d'Italia, ovvero il "Sig. Mario Rossi". Quindi, sebbene saremo sempre raggiungibili in Second Life (non siamo anonimi), possiamo stare tranquilli che nella vita reale nessuno ci verrà a disturbare (siamo in incognito).

IL TOCCO

Il primo esempio che affrontiamo è il più semplice da realizzare. Creiamo il nostro solito cubo di legno e poniamo al suo interno il seguente codice (contenuto nel file *Detection Tocco.lsl*):

```
default
{

 // Evento scatenato dal tocco dell'avatar
 touch_start(integer total_number)
 {

  // Individuiamo la chiave dell'avatar
 // che tocca l'oggetto
  key avatarKey = llDetectedKey(0);

 // Individuiamo il nome
 // dell'avatar che tocca l'oggetto

  string avatarName = llDetectedName(0);
  // Inviamo alcuni messaggi
 // all'avatar che tocca l'oggetto
```

```
  llInstantMessage(avatarKey, avatarName + " mi
                                   hai toccato!");

  // Convertiamo la chiave dell'avatar in stringa
  string savatarKey = (string)avatarKey;
  llInstantMessage(avatarKey, avatarName + " la
                       tua chiave e': " + savatarKey);
 }

// Scatenato ogni volta che
// l'oggetto viene creato dall'inventario
 on_rez(integer start_param)
 {

  // Resetta lo script
  llResetScript();
 }
}
```

Per l'oggetto di cui sopra, implementiamo il solo evento di reazione al tocco dell'avatar (*touch_start*).
Al suo interno incontriamo le prime funzioni di libreria per la "detection": *llDetectedKey* ed *llDetectedName*. Ad entrambe le funzioni passiamo un parametro che rappresenta l'indice dell'oggetto individuato. In questo caso passiamo il valore "zero" (0) in quanto il tocco scaturisce da un solo avatar (vedremo più avanti l'uso di questo indice).
La prima delle due funzioni restituisce la chiave univoca dell'avatar, quindi un valore di tipo *key*; la seconda restituisce il nome ed il cognome (in Second Life) dell'avatar. Nella prova che stiamo effettuando mandiamo semplicemente dei messaggi all'avatar che tocca l'oggetto ed il risultato è visibile in Figura 20Questa tecnica è molto semplice da implementare, ma presuppone un intervento da parte dell'avatar che, per essere individuato, deve esplicitamente toccare il nostro oggetto. Se vogliamo vedere la cosa da un altro punto di vista, possiamo però dire che una tecnica del genere è l'ideale per quelle situazioni in cui, in effetti, vogliamo che l'avatar sia consapevole della sua individuazione che, tra l'altro, potrebbe corrispondere ad una successiva registrazione del suo accesso.Le successive due tecniche che andremo ad analizzare nel prossimo numero della rivista, non necessitano di nessuna interazione esplicita da parte degli avatar e quindi sono tecniche di individuazione "silenti".

CONCLUSIONI

In questa seconda puntata dedicata al mondo di Second Life e del Linden Scripting Language, abbiamo affrontato alcuni argomenti avanzati.
Abbiamo innanzitutto esplorato il processo di creazione degli oggetti che in gergo prende il nome di rez. Abbiamo visto che ogni oggetto creato è univocamente identificabile (ed individuabile) tramite una chiave univoca o UUID (Universal Unique IDentifier); abbiamo visto quanto sia "delicata" questa chiave e cosa succede se inseriamo od estraiamo gli oggetti dal nostro inventario.
Nella seconda parte dell'articolo, abbiamo invece iniziato a trattare le tecniche di "detection", ovvero le metodologie che ci consentono di "tracciare" gli avatar che entrano nelle nostre isole. Abbiamo visto la prima tecnica legata al tocco di una primitiva, la sua implementazione ed il suo significato in ternini di consapevolezza da parte dell'avatar di interagire con i nostri oggetti.
Nella prossima puntata affronteremo le altre tecniche di "detection" ed impareremo ad individuare i nostri visitatori in modo "silente". In particolare affronteremo il problema dei "sensori". Vedremo cosa sono e come usarli all'interno dei nostri script di programmazione per Second Life.

Oscar Peli

Figura 2: Individuiamo l'avatar che tocca il nostro oggetto.

LE SANDBOX

Per i novizi, o comunque per coloro che non posseggono un proprio terreno in cui fare le prove, esistono delle zone franche, dette sandbox, in cui possiamo creare oggetti e provare i nostri script. Una di queste sandbox è raggiungibile al seguente indirizzo: Volksland 9, 248, 21 (gli oggetti che creiamo vengono posti nella cartella "Lost And Found" del nostro Inventory, dopo alcuni minuti di utilizzo al fine di mantenere la sandbox "pulita"; per continuare le nostre prove basta trascinare nuovamente l'oggetto sul terreno).

USARE LE PROPRIETÀ ESTESE IN SQL

UTILIZZIAMO ALCUNE PROPRIETÀ POCO CONOSCIUTE DEL DB DI MICROSOFT. SEMPLICEMENTE STRUTTURANDO IL DATABASE IN MODO ADEGUATO RIUSCIREMO A FACILITARE DI MOLTO LA REALIZZAZIONE DELLE FORM PER L'INPUT/OUTPUT DEI DATI

Un'attività molto frequente, per chi lavora con i database, è la presentazione dei dati in moduli compilabili da parte dell'utente (le famose Forms) o in tabelle per presentare i dati di riepilogo (le Views).
Prendiamo ad esempio una tabella del famoso database di test per SQL Server, il NorthWind, che simula un catalogo reale di un'azienda.
Poniamo il caso di dover creare una maschera di compilazione per la tabella categorie (chiamata Categories nel database).
Tipicamente procederemo disegnando nella Form delle Labels per indicare il nome dei campi con un corrispondente controllo input (textbox, checkbox ecc...) per rappresentare i dati.
Nell'immagine 1 possiamo vedere infatti le due label correlate ai rispettivi input per i dati.

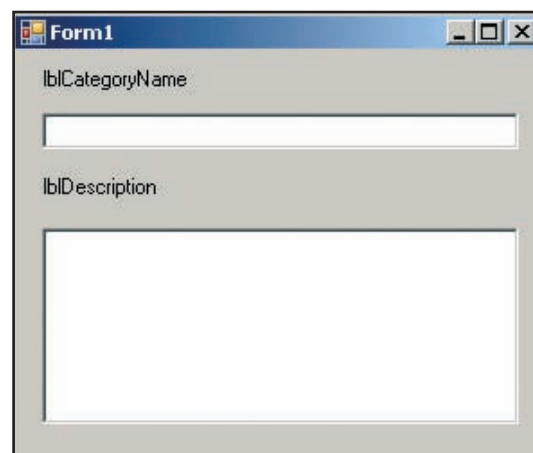



Fig. 1: una form per la gestione dei dati

Naturalmente spesso la Form è ben più complessa ma qui, a titolo di esempio, abbiamo voluto prendere un caso molto semplice.
L'operazione di assegnazione di un nome all'etichette (controlli Label) è sicuramente banale, ma proprio in quanto tale anche molto noiosa (occorre andare nelle proprietà del controllo e scrivere il testo ecc...).
Per semplificarci la vita ci viene in soccorso una caratteristica spesso misconosciuta e poco sfruttata di SQL Server: l'utilizzo delle proprietà estese.
In SQL Server, fin dalla versione 2000, è infatti possibile associare a tabelle, viste, stored procedure, colonne (di tabelle e di viste) delle proprietà aggiuntive chiamate *proprietà estese* (*Extended Properties*).
Il tool di gestione dei database *Microsoft SQL Server Management Studio* (disponibile anche nella versione gratuita *Express*) offre la possibilità di impostare visivamente le *proprietà estese*, per farlo è sufficiente cliccare con il tasto destro sull'oggetto (tabella, vista, colonna ecc...) e selezionare la voce *properties*, apparirà una maschera dove è possibile aggiungere o modificare le proprietà estese connesse all'oggetto (figura 2).

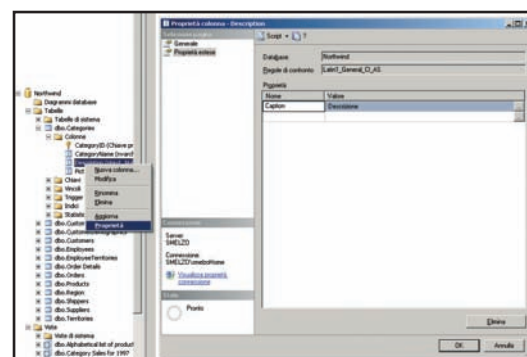



Fig. 2: gestione visuale delle proprietà estese

RECUPERO DELLE PROPRIETÀ ESTESE

Una volta capito come impostare le proprietà estese vediamo come recuperarne il valore.
Il recupero di questi metadati avviene con una normale Select effettuata però sulla funzione di sistema *fn_listextendedproperty* :

REQUISITI

Conoscenze richieste
SQL Server e .NET

Software
SQL Server anche in versione Express

Impegno

Tempo di realizzazione

```
SELECT   *
FROM   ::fn_listextendedproperty ('caption', 'user',
         'dbo', 'table', 'Categories', 'column', null)
```

Gli argomenti di questa funzione sono:

1. Nome della proprietà da recuperare (o il valore NULL per recuperarle tutte)
2. Tipo di primo livello (in pratica la parola 'user')
3. Nome del tipo di primo livello al quale è associata la proprietà (molto probabilmente 'dbo')
4. Tipo di secondo livello ('table', 'view' ecc...) al quale è associata la proprietà
5. Nome del tipo di secondo livello (il nome della tabella o vista, nel nostro caso 'Categories')
6. Tipo di terzo livello ('column', 'parameter' ecc...) al quale è associata la proprietà
7. Nome del tipo di terzo livello

In pratica, a parte il primo argomento che è il nome della proprietà, gli altri sono coppie di argomenti collegati che servono a collocare la posizione delle proprietà da recuperare.
Nel nostro caso quindi è come se avessimo detto : recupera la proprietà "*caption*" che appartiene allo *user dbo* nella *table Categories* in tutte le colonne (da notare l'uso di NULL, o della parola DEFAULT, per recuperare tutti i valori).

Il risultato della query, nel nostro caso, sarà:

objtype	objname	name	value
COLUMN	CategoryName	Caption	Nome categoria
COLUMN	Description	Caption	Descrizione

Ovvero, rifinendo ulteriormente la query come:

```
SELECT   objname as field, value as caption
FROM   ::fn_listextendedproperty ('caption', 'user',
         'dbo', 'table', 'Categories', 'column', null)
```

Otterremo:

field	caption
CategoryName	Nome categoria
Description	Descrizione

In questo modo i metadati sono più chiari, tuttavia il nostro obiettivo è quello di inserire nella lista tutti i campi della tabella, quelli che hanno una *caption* impostata ed anche gli altri.
La query quindi sarà un JOIN tra i metadati delle colonne (forniti dalla vista di sistema *INFORMATION_SCHEMA.COLUMNS*) e *fn_listextendedproperty* :

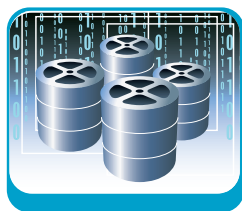
```
SELECT
info.COLUMN_NAME as field,
isnull(exProp.value,info.COLUMN_NAME) as caption
FROM  INFORMATION_SCHEMA.COLUMNS as info
left outer join ::fn_listextendedproperty ('caption',
'user', 'dbo', 'table', 'Categories', 'column', default) as
                                                    exProp
ON info.COLUMN_NAME = exProp.objname
WHERE info.TABLE_NAME = 'Categories'
```

Il risultato sarà proprio la tabella con nomi delle colonne e, dove c'è, la caption (altrimenti si ripete il nome):

field	caption
CategoryID	CategoryID
CategoryName	Nome categoria
Description	Descrizione
Picture	Picture

UTILIZZO DELLE PROPRIETÀ ESTESE

Torniamo adesso, per un attimo, alla form che abbiamo visto nella figura 1; nel progetto impostiamo una semplice funzione che ci consente di recuperare i metadati di una tabella in forma di NameValueCollection (un insieme composto da coppie chiave/valore):

```
Private Function GetTableInfo(ByVal tableName As
            String, ByVal propertyName As String) As
                              NameValueCollection
'Stringa di connessione
Dim strConn As String = "Data
                          Source=Localhost;Initial
      Catalog=Northwind;user=nwuser;password=nw"
Dim cnn As New SqlConnection(strConn)
'Composizione Query
  Dim cmdText As New System.Text.StringBuilder
      cmdText.AppendLine("SELECT ")
      cmdText.AppendLine("info.COLUMN_NAME as
                                         field,")
cmdText.AppendLine("isnull(exProp.value,info.COLUM
                               N_NAME) as caption")
      cmdText.AppendLine("FROM
          INFORMATION_SCHEMA.COLUMNS as info")
      cmdText.AppendLine("left outer join")
      cmdText.AppendLine("
       ::fn_listextendedproperty (@PROPERTYNAME,")
```

```
        cmdText.AppendLine("'user', 'dbo', 'table',
                                        @TABLENAME,")
        cmdText.AppendLine("'column', default) as
                                              exProp")
        cmdText.AppendLine("ON info.COLUMN_NAME
                                    = exProp.objname")
        cmdText.AppendLine("WHERE info.TABLE_NAME
                                       = @TABLENAME")
'Creazione Command e impostazione parametri
        Dim cmd As New
                    SqlCommand(cmdText.ToString, cnn)
cmd.Parameters.AddWithValue("@PROPERTYNAME",
                                        propertyName)
        cmd.Parameters.AddWithValue("@TABLENAME",
                                           tableName)
        cnn.Open()
'Esecuzione Command e recupero Reader
        Dim reader As SqlDataReader =
                                   cmd.ExecuteReader
                  (CommandBehavior.CloseConnection)
'Creazione e riempimento della Collection
        Dim result As New NameValueCollection
        While reader.Read
            Dim name As String = reader.GetString(0)
            Dim caption As String = reader.GetString(1)
            result(name) = caption
        End While
        reader.Close()
        Return result
End Function
```

Da notare che la funzione servirà a recuperare le proprietà estese corrispondenti ad un determinato nome (identificato dall'argomento *propertyName*) associate alle colonne di una tabella (identificata dall'argomento *tableName*) e quindi ha un uso generale.
La funzione viene utilizzata nell'evento Load della pagina :

```
Private Sub frmTest_Load(ByVal sender As
          System.Object, ByVal e As System.EventArgs)
                                  Handles MyBase.Load
        Dim tableInfo As NameValueCollection =
                 GetTableInfo("Categories", "Caption")
        For Each key As String In tableInfo.Keys
            Dim lblName As String = "lbl" & key
            Dim ctrl As Control =
                              Me.Controls.Item(lblName)

            If ctrl IsNot Nothing Then
                If TypeOf ctrl Is Label Then
                    CType(ctrl, Label).Text =
                                         tableInfo(key)
                End If
            End If
        Next
End Sub
```

In pratica la procedura cerca un controllo denominato "lbl" + <nome del campo> e, se lo trova (e se è di tipo Label), imposta la proprietà Text con la Caption recuperata dalle proprietà estese.

DETTAGLI SULLE PROPRIETÀ ESTESE

Abbiamo visto un utilizzo piuttosto semplice delle proprietà estese ma, prima di vedere altre applicazioni, è opportuno ritornare sui dettagli tecnici di questa caratteristica di SQL Server. Le proprietà estese, come abbiamo visto, sono informazioni aggiuntive che possono essere collegate agli oggetti del database. Ogni proprietà estesa è un valore di tipo *sql_variant* e può contenere fino a 7500 byte di dati. Ai fini delle proprietà estese gli oggetti del database SQL Server sono classificati in tre livelli:

- Livello 0 – Utente, Tipo di dati definito dall'utente
- Livello 1 – Tabella, Vista, Stored Procedure, Funzione, Regola, Valore predefinito
- Livello 2 - Colonna, indice, vincolo, parametro e trigger

I riferimenti ad un oggetto di un certo livello debbono essere qualificati con i nomi degli oggetti di livello. Ad esempio, se ci riferiamo ad una colonna (livello 2) occorre specificare anche la tabella (livello 1) che contiene la colonna e l'utente (livello 0) a cui appartiene la tabella. A livello di Transact SQL le extended properties sono gestite con tre Stored Procedure ed una Funzione:

- **sp_addextendedproperty** - Aggiunge una nuova proprietà estesa ad un oggetto di database.
- **sp_updateextendedproperty** - Aggiorna il valore di una proprietà estesa.
- **sp_dropextendedproperty** - Elimina una proprietà estesa esistente.
- **fn_listextendedproperty** – Recupera il valore di proprietà estese.

AGGIUNGERE UNA PROPRIETÀ ESTESA

La procedura sp_addextendedproperty di sistema consente di aggiungere una nuova proprietà estesa ad un oggetto di database, i parametri sono: nome e valore della proprietà

NOTA

DOCUMENTAZIONE

La fonte primaria di documentazione sulle Extended Properties è la documentazione di SQL Server (Book OnLine) disponibile con ogni installazione di SQL Server anche in versione Express o nel sito http://msdn2.microsoft.com/en-us/library

seguiti da tre coppie tipo/nome che servono a localizzare l'oggetto.
Ad esempio, per aggiungere la proprietà "Caption" con il valore "Descrizione" della colonna "Description" della tabella "Categories" del database potremmo scrivere:

```
EXEC   sp_addextendedproperty 'Caption',
       'Descrizione', 'user', 'dbo', 'table', 'Categories',
                                   'column', 'Description'
```

Quando si tratta di impostare proprietà estese per la tabella (livello 1) o l'utente (livello 0) le coppie di parametri tipo/nome corrispondenti ai livelli inferiori sono impostate a *null*.
Vediamo ad esempio come aggiungere la proprietà "Title" con il valore "Categorie" alla tabella "Categories":

```
EXEC   sp_addextendedproperty 'Title', 'Categorie',
        'user', 'dbo', 'table', 'Categories', NULL, NULL
```

MODIFICARE UNA PROPRIETÀ ESTESA

La procedura sp_updateextendedproperty di sistema consente di modificare una proprietà estesa di un oggetto di database.
L'utilizzo è simile a *sp_addextendedproperty*, i parametri sono: nome e valore della proprietà seguiti da tre coppie tipo/nome che servono a localizzare l'oggetto.
Ad esempio per modificare la proprietà "Caption" con il valore "Dettagli" della colonna "Description" della tabella "Categories" del database potremmo scrivere:

```
EXEC   updateextendedproperty 'Caption', 'Dettagli',
          'user', 'dbo', 'table', 'Categories', 'column',
                                           'Description'
```

CANCELLARE UNA PROPRIETÀ ESTESA

La procedura sp_dropextendedproperty di sistema consente di cancellare una proprietà estesa di un oggetto di database. I parametri sono quelli visti per le altre stored procedure salvo che, ovviamente, non viene specificato il valore della proprietà da cancellare, ma solo il nome. Così, ad esempio, per cancellare la proprietà "Caption" della colonna "Description" della tabella "Categories" del database potremmo scrivere:

```
EXEC sp_dropextendedproperty 'Caption', 'user',
       'dbo', 'table', 'Categories', 'column', 'Description'
```

RECUPERARE LE PROPRIETÀ ESTESE

La funzione fn_listextendedproperty di sistema consente di recuperare i valori delle proprietà estese degli oggetti di database.
I parametri sono: nome della proprietà seguito da tre coppie tipo/nome che servono a localizzare l'oggetto. La tabella restituita dalla funzione è nel formato:
Quindi, per recuperare le proprietà "Caption" associate a tutte le colonne della tabella "Categories" scriveremo:

```
SELECT   * FROM   ::fn_listextendedproperty
 ('Caption', 'user', 'dbo', 'table', 'Categories', 'column',
                                                     NULL)
```

Nome colonna	Tipo di dati
objtype	sysname
objname	sysname
name	sysname
value	sql_variant

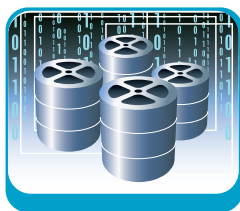
LE PROPRIETÀ ESTESE IN SCENARI PIÙ COMPLESSI

L'utilizzo che abbiamo visto delle proprietà estese (per le *Caption*) è abbastanza semplicistico. Spostiamoci adesso sul campo delle applicazioni web per vedere qualcosa di più complesso. Si tratta qui di ottenere una tabella in formato XML da utilizzare per l'elaborazione con AJAX. L'XML conterrà, in questo caso sia i metadati (nome del campo, caption e tipo di dati) che le righe dei dati veri e propri. Ciò si rivelerà particolarmente utile per comporre la tabella HTML usando AJAX.
Impostiamo quindi, in Visual Studio o altro IDE per .NET, un nuovo progetto web e, nella cartella App_Code definiamo una classe, XmlHandler, che ha il compito di restituire i dati in formato XML. Nel corpo della classe definiamo i campi e le proprietà private insieme al costruttore:

```
Private ConnectionString As String
Private ObjName As String
Private WhereAndOrderStatements As String
Private ReadOnly Parameters() As SqlParameter
Private _ExPropertyName As String = "Caption"
              'nome di default per la proprietà estesa
Public Property ExPropertyName() As String
```

NOTA

DIETRO LE QUINTE

Nell'articolo vediamo come gestire le proprietà estese con gli strumenti standard di SQL , è però utile sapere che, in fin dei conti, le proprietà estese sono conservate in una comune tabella di sistema chiamata sysproperties lo stesso risultato ottenuto con la funzione fn_listextendedpropert y, è perciò ottenibile anche con una query che combini sysproperties con la tabella sysobjects.

```
        Get
            Return _ExPropertyName
        End Get
        Set(ByVal Value As String)
            _ExPropertyName = Value
        End Set
    End Property
    Private _FieldList As String = "*" 'lista dei campi
                                da estrarre, default * (tutti)
    Public Property FieldList() As String
        Get
            Return _FieldList
        End Get
        Set(ByVal Value As String)
            _FieldList = Value
        End Set
    End Property
    Sub New(ByVal strConn As String, ByVal
                    tableOrViewName As String, ByVal
            whereAndOrderStatements As String, ByVal
            ParamArray parameters() As SqlParameter)
        Me.ConnectionString = strConn
        Me.ObjName = tableOrViewName
        Me.WhereAndOrderStatements =
                            whereAndOrderStatements
        Me.Parameters = parameters
    End Sub
```

Notiamo che nel costruttore di possono impostare :

1. la stringa di connessione
2. il nome della tabella o della vista
3. eventuali comandi T-SQL per definire clausole WHERE e/o ORDER BY
4. eventuali parametri aggiuntivi che l'utente può definire nell'argomento precedente

La lista dei campi da estrarre (*FieldList*) ed il nome della proprietà estesa (*ExPropertyName*) sono definite invece come proprietà pubbliche impostabili successivamente alla costruzione dell'oggetto in quanto hanno comunque valori di default ("*", ovvero tutti i campi, per *FieldList* e "Caption" come *ExPropertyName*). Passiamo adesso a definire le due funzioni che creano l'istruzione T-SQL da passare ad un SQL Command. La prima (*GetMetadataStatement*) imposta una SELECT sui metadati della tabella o vista mentre la seconda (*GetSqlStatement*) imposta la SELECT di estrazione dei dati veri e propri:

```
    Private Function GetMetadataStatement() As
                                            String
        Dim sb As New System.Text.StringBuilder
        sb.AppendLine("SELECT ")
        sb.AppendLine("info.COLUMN_NAME as field,")
sb.AppendLine("isnull(exProp.value,info.COLUMN_NA
                                    ME) as caption,")
        sb.AppendLine("info.DATA_TYPE as datatype")
        sb.AppendLine("FROM
            INFORMATION_SCHEMA.COLUMNS as info")
        sb.AppendLine("left outer join")
        sb.AppendLine(" ::fn_listextendedproperty
                                    (@PROPERTYNAME,")
        sb.AppendLine("'user', 'dbo', 'table',
                                        @TABLENAME,")
        sb.AppendLine("'column', default) as exProp")
        sb.AppendLine("ON info.COLUMN_NAME =
                                    exProp.objname")
        sb.AppendLine("WHERE info.TABLE_NAME =
                                        @TABLENAME")
        Return sb.ToString
    End Function
    Private Function GetSqlStatement() As String
        Dim sb As New System.Text.StringBuilder
        sb.AppendLine(GetMetadataStatement)
                        'aggiunge la SELECT sui metadati
        sb.AppendFormat("SELECT {0} FROM [{1}]",
                Me.FieldList, Me.ObjName).AppendLine()
        sb.AppendLine(Me.WhereAndOrderStatements)
        Return sb.ToString
    End Function
```

Definiamo adesso la funzione che restituisce il Command SQL da utilizzare per l'estrazione dei dati:

```
    Private Function GetSqlCommand() As
                                        SqlCommand
        Dim cnn As New
                SqlConnection(ConnectionString)
        Dim cmd As New
            SqlCommand(Me.GetSqlStatement, cnn)
        'aggiunta parametri per i metadati
cmd.Parameters.AddWithValue("@PROPERTYNAME",
                                Me.ExPropertyName)
        cmd.Parameters.AddWithValue("@TABLENAME",
                                        Me.ObjName)
        'aggiunta altri parametri definiti dall'utente
        For Each param As SqlParameter In
                                    Me.Parameters
            cmd.Parameters.Add(param)
        Next
        Return cmd
    End Function
```

Ed infine impostiamo il metodo pubblico che restituisce il documento XML:

```
    Public Function GetXml() As XmlDocument
        Dim result As New XmlDocument
        Dim root As XmlElement =
result.AppendChild(result.CreateElement("response"))
```

```
    Dim meta As XmlElement =
    root.AppendChild(result.CreateElement("meta"))
    Dim rows As XmlElement =
    root.AppendChild(result.CreateElement("rows"))
    Dim cmd As SqlCommand =
                                        Me.GetSqlCommand
    cmd.Connection.Open()
    Dim reader As SqlDataReader =
cmd.ExecuteReader(CommandBehavior.CloseConnecti
                                                on)
    'riempie il nodo <meta/> leggendo la query sui
                                          metadati
    While reader.Read
        'crea <column>
        Dim column As XmlElement =
  meta.AppendChild(result.CreateElement("column"))
  column.SetAttribute("field", reader.GetString(0))
  column.SetAttribute("caption",reader.GetString(1))
 column.SetAttribute("datatype",reader.GetString(2))
    End While
    'riempie le righe con i dati

    If reader.NextResult Then
        While reader.Read
            Dim row As XmlElement =
    rows.AppendChild(result.CreateElement("row"))

            Dim fieldCount As Integer =
                                   reader.FieldCount
            For i As Integer = 0 To fieldCount - 1
                Dim field As XmlElement =
    row.AppendChild(result.CreateElement("field"))
                Dim fieldName As String =
                                   reader.GetName(i)
                field.SetAttribute("name", fieldName)
                Dim fieldValue As String = ""
                If Not reader.IsDBNull(i) Then
                    fieldValue =
                          reader.GetValue(i).ToString()
                End If
                field.InnerText = fieldValue
            Next
        End While
    End If
    reader.Close()
    Return result
End Function
```

Notiamo la costruzione del documento effettuata con i metodi del DOM di .NET e i due cicli While sull'oggetto Reader: il primo che legge i risultati della query sui metadati ed il secondo che legge i dati veri e propri.
A questo punto la nostra classe è sufficientemente generica ed adattabile a più circostanze; poiché a noi serve restituire un output XML non occorre una pagina ASPX, è sufficiente un gestore generico ASHX che impostiamo in questo modo nella Root del web site:

```
<%@ WebHandler Language="VB" Class="Test" %>
Imports System
Imports System.Web
Imports System.Xml
Public Class Test : Implements IHttpHandler
    Public Sub ProcessRequest(ByVal context As
                        HttpContext) Implements
                        IHttpHandler.ProcessRequest
        Dim strConn As String = "Data
                          Source=Localhost;Initial
      Catalog=Northwind;user=nwuser;password=nw"
        Dim xmlHnd As New XmlHandler(strConn,
                                "Categories", "")
        Dim responseDoc As XmlDocument =
                                     xmlHnd.GetXml
        context.Response.ContentType = "text/xml"
        responseDoc.Save(context.Response.Output)
    End Sub
    Public ReadOnly Property IsReusable() As Boolean
                  Implements IHttpHandler.IsReusable
        Get
            Return False
        End Get
    End Property
End Class
```

Per testare l'output sarà quindi sufficiente aprire nel browser il file del gestore (che noi abbiamo chiamato test.ashx) per vedere i risultati. Se tutto è andato bene l'output sarà qualcosa di simile a quello evidenziato in figura 4.

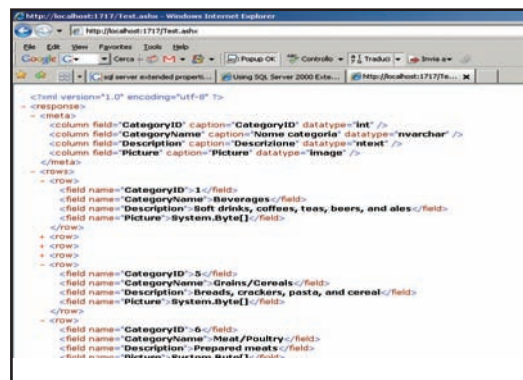
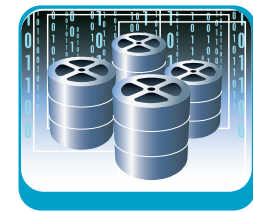



Fig. 2: gestione visuale delle proprietà estese

La trasformazione di questi dati in HTML attraverso AJAX non rientra negli scopi del nostro articolo, comunque abbiamo riportato un esempio completo che sicuramente vi illustrerà come gestire questo tipo di scenario

Francesco Smelzo

068-076:032-035 1-10-2007 17:13 Pagina 68

IL TUO PROSSIMO LINGUAGGIO

CON TANTI LINGUAGGI DI PROGRAMMAZIONE IN GIRO, PUÒ ESSERE DIFFICILE DECIDERE SU QUALE CONCENTRARSI. CHE TU SIA UN ESPERTO O UN PRINCIPIANTE, QUESTO ARTICOLO PUÒ AIUTARTI A SCEGLIERE IL PROSSIMO

Forse ti sei stufato del linguaggio che usi ogni giorno da anni, e vuoi imparare qualcosa di nuovo. Oppure hai deciso che imparare un nuovo linguaggio ogni tanto rassoda i tuoi muscoli informatici. O magari sei nuovo nel mondo della programmazione, e devi scegliere il tuo primo linguaggio. In ogni caso, la scelta può sembrare difficile. Di linguaggi ce ne sono a bizzeffe. La Wikipedia ne elenca oltre cinquecento, senza contare gli innumerevoli dialetti. Alcuni sono celeberrimi e richiestissimi dalle aziende, altri sono noti a stento ai propri autori. Alcuni sono seri strumenti di sviluppo per applicazioni critiche, altri semplici giocattoli praticamente inutili. Se si sfronda la lista e ci si concentra sui “linguaggi che contano”, ne restano sì e no una ventina. Se si eliminano quelli più specifici, o quelli che hanno già passato da tempo i propri anni migliori, la scelta si riduce a mezza dozzina linguaggi alla moda e un pugno di vecchie glorie. Alcuni tra questi sono linguaggi ideali per iniziare, altri sono i più richiesti dalle aziende, altri ancora sono i migliori per allargare i propri orizzonti di programmatore. Diamo uno sguardo alle star della programmazione (e a qualche astro nascente), per decidere quale sarà il prossimo ospite della tua cassetta degli attrezzi.

JAVA

Quando venne concepito, nei primi anni '90, doveva essere un linguaggio per elettrodomestici e cellulari. Non ci riuscì. Poi arrivò Internet, e si propose come linguaggio per il Web. Anche questo fu un fallimento. Poi tentò di imporsi come linguaggio applicativo sui desktop. Niente da fare.
Ma allora, come mai Java è oggi il linguaggio di programmazione più popolare e richiesto? La risposta è sui server. Quello che all'inizio veniva spesso liquidato come un giocattolo lento e inaffidabile è diventato la base dei sistemoni aziendali che scodellano pagine web, transazioni bancarie, informazioni finanziarie...
L'asso nella manica di Java è stata la sua “macchina virtuale”. Il compilatore Java non genera codice nativo per una determinata macchina come fa il C++. Il computer che fa girare Java è a sua volta un programma, la Java Virtual Machine. Quindi lo stesso programma Java può girare su qualsiasi computer abbia una JVM – il che quasi certamente vuol dire che gira anche sul tuo. Negli anni '90, mentre Internet e Linux rompevano le barriere tra i sistemi operativi, questa caratteristica è stata essenziale.
La macchina virtuale non è una buona idea solo per questioni di compatibilità. Costruisce anche una rete di sicurezza che ha fatto di Java un linguaggio sorridente in un mondo di linguaggi ringhiosi. Alcuni tra i più orrendi bug del C++, come gli errori nella gestione della memoria, sono semplicemente impossibili in Java. Così Java ha reso la programmazione a oggetti accessibile a tutti, e questa è stata forse la sua più grande conquista.

PERCHÉ IMPARARLO

Java è la scelta più ovvia per chi con il computer ci lavora. A parte il mondo Microsoft, non esistono ancora alternative valide a Java per buona parte dei grandi progetti. Le risorse su Internet, comprese le librerie open source e gli IDE come Eclipse e NetBeans, sono innumerevoli, di altissima qualità, e quasi sempre belle gratis.

PERCHÉ IGNORARLO

Da molto tempo, ormai, Java non è più un linguaggio “fico”. Il giovane ribelle va verso una sonnacchiosa mezza età. E se è vero che la domanda delle aziende resta alta, è anche vero che tutti, compresi mia nonna e il mio salumiere, si vendono ormai come programmatori Java.

IL PRIMO LINK

Thinking in Java di Bruce Eckel, uno dei libri migliori per imparare Java e la programmazione a oggetti. Potete scaricarlo gratis da http://www.mindview.net/Books/TIJ

C#

A Microsoft, questa faccenda di Java non è mai andata giù. Ma come? Tutta questa fatica per dominare il mercato con Windows, e improvvisamente arriva un linguaggio che tratta il sistema operativo con spocchiosa indifferenza? E' per questo che nel 2001, l'aziendona di Zio Bill pubblica la sua alternativa a Java: un linguaggio quasi identico a Java, con una Virtual Machine come Java (parte di un sistema che Microsoft ha chiamato .NET), ma inizialmente solo per Windows.
Che C# sia un clone di Java non c'è dubbio, ma a volte l'allievo supera il maestro. Mentre Java si trascina tra lungaggini politiche, C# non esita a sperimentare funzionalità nuove; e mentre Java si preoccupa della propria fama di linguaggio "serio", C# non si vergogna di semplificare in campi essenziali come le interfacce grafiche e i database. Alla fine è spesso Java che deve adeguarsi, seguendo con un po' di affanno la via tracciata dal rivale. E grazie a progetti open source come Mono è oggi possibile, con qualche sacrificio, far girare C# su sistemi diversi da Windows.

PERCHÉ IMPARARLO

In generale è un po' più produttivo di Java, almeno per i piccoli progetti. E magari la vostra azienda preferisce roba di Microsoft, o volete ritagliarvi una nicchia in un mercato saturo di javisti.

PERCHÉ IGNORARLO

È come Java, ma senza il dinamico mondo open source che circonda Java. La gran parte dei progetti interessanti nel mondo C# sono conversioni di progetti Java. E quando si tratta di compatibilità multi-piattaforma, Java resta inarrivabile.

IL PRIMO LINK

La guida ufficiale di Microsoft su http://msdn2.microsoft.com/en-us/vcsharp.

NOTA

OGGETTI O NON OGGETTI

Quando sceglie l'automobile, un bravo acquirente chiede al concessionario: è sicura? Allo stesso modo, un bravo programmatore che vuole scegliere un nuovo linguaggio dovrebbe sempre chiedere: è orientato agli oggetti? Se non sapete cos'è la programmazione a oggetti, state tranquilli. Studiando il vostro prossimo linguaggio probabilmente lo imparerete. Banalizzando, possiamo dire che tanto più un linguaggio è "orientato agli oggetti", tanto più è moderno e potente.

PHP

Qual è il modo più veloce per produrre pagine web dinamiche? Un modo ovvio è usare dei "template": pagine HTML che contengono codice, che a sua volta genera altri pezzi di HTML. Il server che riceve una richiesta esegue il codice nella pagina, e il risultato viene inviato al client. Molti hanno implementato questa idea, compresi linguaggi come ASP di Microsoft, o tecnologie come JSP di Sun. Ma nessuno, forse, lo ha fatto bene come PHP.
PHP è un linguaggio del 1996, creato da un programmatore nordeuropeo che voleva usarlo come sostituto per sostituire gli script Perl del proprio sito (il nome significava in origine "Personal Home Page"). Negli anni, è diventato uno dei più popolari linguaggi per il web.
Il problema fondamentale di PHP è che, come un Visual Basic del Web, invita i programmatori a mescolare interfaccia e logica. E visto che la maggior parte dei siti dinamici accedono ad un database, il codice PHP contiene spesso anche una bella quantità di chiamate SQL. Il risultato per chi non ha molta disciplina è una massa di codice inestricabile e spaghettiforme. Anche per questo la fama di PHP appare ultimamente un po' appannata, e la gente si rivolge verso tecnologie alternative come Ruby on Rails o Django.

PERCHÉ IMPARARLO

È ancora molto usato, ed è ancora uno dei modi più veloci per mettere insieme un sito web.

PERCHÉ IMPARARLO

I programmatori PHP sono tanti, ma di solito non sono tenuti in grande considerazione. E oggi l'attenzione di tutti è concentrata su altro.

IL PRIMO LINK

Il tutorial di W3Schools, su http://www.w3schools.com/php/.

VISUAL BASIC .NET

Proprio come la guerra mondiale, Visual Basic ha cambiato il mondo. Quando venne introdotto da Microsoft nel 1991, cambiò la percezione della programmazione: non più un'arte arcana riservata a pochi, ma un modo accessibile a tutti per mettere insieme velocemente un'applicazione.
I risultati non sono sempre stati positivi. Innumerevoli aziende di software hanno scelto Visual Basic per scrivere importanti applicazioni gestionali, molte delle quali oggi scricchiolano in attesa di una riscrittura. Visual Basic è ideale per mettere insieme quattro pulsanti e due tabelle, ma l'opinione comune è che non sia un linguaggio che incoraggia il programmatore a progettare applicazioni solide.
Nel 2002 Microsoft ha introdotto Visual Basic .NET, un'evoluzione di VB che gira sulla stessa macchina virtuale del più sofisticato C#. Nonostante la carriolata di nuove caratteristiche, molti programmatori Visual Basic sono inorriditi di fronte alle numerose e fondamentali incompatibilità con il "vecchio" VB. Il risultato è stata una migrazione di massa verso altri linguaggi, e oggi Visual Basic sembra finalmente sul viale del tramonto.

PERCHÉ IMPARARLO

Solo se avete usato per anni il vecchio Visual Basic, e proprio non vi va di perdere tempo a imparare qualcosa di nuovo.

PERCHÉ IGNORARLO

Anche se C# è più complesso di Visual Basic, vale la pena di concentrarsi su quello: fa le stesse cose, e le fa meglio.

IL PRIMO LINK

La guida ufficiale di Microsoft è disponibile su http://msdn2.microsoft.com/vbasic.

JAVASCRIPT

Quando fu creato da Netscape nel 1995, sembrava quasi un giocattolo: un linguaggio script che gira direttamente nel browser, JavaScript era per i client quello che PHP era per i server. La gente lo usava per inserire piccoli effetti e pezzettini di codice nelle pagine web, e pareva che non ci fosse altro motivo per usarlo. Anche il nome e la sintassi ispirata a Java erano solo mossa di marketing per imporre un linguaggino che con il più robusto Java non aveva proprio niente a che vedere.
Poi successe che tra tante incompatibilità e guerre tecnologiche, JavaScript rimase praticamente l'unico modo affidabile e sicuro per far girare codice nel browser. Quando qualcuno pensò di mescolare JavaScript con un po' di XML e una manciata di trucchetti tecnologici, il risultato fu un frankenstein di tecnologie di nome AJAX, che permetteva finalmente di costruire interfacce decenti per le applicazioni web. Così, mentre i linguaggi più blasonati restano al palo, JavaScript è il linguaggio in più rapida espansione del momento. Nessuno sa quanto durerà, ma un quarto d'ora (o qualche annetto) di celebrità non glielo leva più nessuno.

PERCHÉ IMPARARLO

Se lavorate sul Web, JavaScript è il linguaggio che può farvi fare il salto dalle interfacce degli anni '90 allo stile del Web 2.0.

PERCHÉ IGNORARLO

Se non programmate per il web, JavaScript è solo un altro linguaggio dinamico, e nemmeno uno dei più flessibili.

IL PRIMO LINK

Il tutorial di W3Schools, su http://www.w3schools.com/js/.

PYTHON

Creato nel 1991 dal "benevolo dittatore" Guido van Rossum, Python (il cui nome non si ispira ai rettili, ma ai comici inglesi Monty Python) ha cambiato le carte in tavola. Prima di Python esistevano i linguaggi "seri", quelli per costruire sistemi, e i "linguaggi di script", buoni al massimo per scrivere semplici programmini di utilità. Python ha dimostrato negli anni che un linguaggio interpretato può essere non solo più produttivo di un Java o di un C++, ma anche più elegante ed espressivo. Python è in un certo senso un linguaggio di nicchia. Le sue nicchie, però, sono talmente tante da renderlo uno dei linguaggi più amati e popolari. Dagli effetti speciali per il cinema ai siti web, dall'industria dell'elettronica all'amministrazione di sistema, Python è un linguaggio che conta eserciti di fan sfegatati. E se un giorno tutto il codice Python dovesse sparire, lo sapremmo tutti subito: il primo ad andare giù sarebbe Google, che lo usa in abbondanza.

PERCHÉ IMPARARLO

Un bel linguaggio di script moderno, utile per i vostri lavoretti quotidiani ma anche per scrivere programmi "seri". Alcuni framework, come Django, meritano attenzione. Ed è sicuramente più veloce di Ruby, il suo più vicino concorrente.

PERCHÉ IGNORARLO

Il cugino Ruby, grazie anche al suo modello object-oriented più "puro", sta accentrando tutta l'attenzione su di sè. E per quanto Python sia sicuramente un linguaggio utile per chiunque, non sono molte le aziende che vi assumeranno perché avete aggiunto un pitone al vostro curriculum.

IL PRIMO LINK

Il libro Dive Into Python, su http://www.diveintopython.org.

RUBY

Ruby è giovanile, ma ha quasi quindici anni. Fino al 2000 è rimasto confinato al Giappone, suo paese di origine. Poi qualcuno in occidente ha scoperto questo linguaggio di script potente, più "orientato agli oggetti" di Python e più elegante di Perl, e Ruby si è ritagliato la sua nicchia di fan. Ma il meglio doveva ancora arrivare.
Nel 2004, un programmatore danese ha pubblicato Ruby on Rails, un framework basato su Ruby per sviluppare siti web. E' stato un terremoto, con seri professionisti che testimoniavano incrementi di produttività fantascientifici, e le comunità di tutti i linguaggi che si affannavano a sviluppare nuovi framework sull'esempio di Rails. Per citare Nathan Torkington, un esperto di open source: "Guardare Ruby on Rails è come guardare uno di quei film di kung-fu dove una dozzina di energumeni si apprestano a malmenare il ragazzino appena arrivato in città, e vengono riempiti di ceffoni in una gran varietà di modi pittoreschi".
E così Ruby, come quegli attori che restano inchiodati al loro ruolo di maggiore successo, è diventato famoso come "il linguaggio di Rails". Ma i suoi appassionati giurano che è solo questione di tempo prima che Ruby conquisti il mondo, o almeno un paio di continenti.

PERCHÉ IMPARARLO

In una parola: Rails. Ma non solo. La comunità Ruby è attualmente il quartiere più fico della città, quello dove gli artisti e i personaggi famosi tirano fuori idee nuove. E Ruby è anche un ottimo linguaggio didattico, ideale per chi impara a programmare per la prima volta.

PERCHÉ IGNORARLO

Qualcuno dice che è solo una moda passeggera. Se preferite un valido concorrente come Python, potete anche aspettare e scoprire se la Ruby-mania passa da sé.

IL PRIMO LINK

Try Ruby, un fantastico minicorso interattivo online: http://tryruby.hobix.com.

I MARKUP LANGUAGES

Ma HTML, sarà un linguaggio? Sì, ma non certo un linguaggio di programmazione. Lo stesso vale per XML, l'altro onnipresente linguaggio di markup. XML, però, è usato spesso come base per "veri" linguaggi, ancorché di dubbia leggibilità. Un esempio particolarmente sgraziato è XSLT, per trasformare dati XML che è a sua volta basato su XML. Ecco Hello World in XSLT:

```
<xsl:stylesheet version="1.0"
 xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="/">
    <xsl:text>Hello World!</xsl:text>
  </xsl:template>
</xsl:stylesheet>
```

Bleah! Ma XSLT è molto utile quando dovete trasformare un file XML, quindi nonostante la sua brutta sintassi potrebbe toccarvi di usarlo. In generale, i markup language sono una buona idea quando si tratta di rappresentare dati. E oggi ci sono alternative più eleganti (anche se meno popolari) ad XML, come YAML o JSON, che si imparano in un attimo e si integrano con tutti i linguaggi più comuni.

PERCHÉ IMPARARLI

Che vi piacciano o meno, HTML, XML e i loro innumerevoli dialetti sono uno strumento essenziale per moltissimi programmatori.

PERCHÉ IGNORARLI

Fatelo solo a vostro rischio.

IL PRIMO LINK

http://www.w3schools.com.

I MORIBONDI

Sempre in bilico tra la vita e la morte, mantenuti in vita da nicchie di conservatori e da vecchi progetti che rifiutano di trapassare dignitosamente: questo è il destino di alcuni linguaggi che dovrebbero ormai appartenere ad un museo. Linguaggi come il Fortran (classe 1954), un dinosauro che il già citato Dijkstra definì una "malattia infantile" e che ancora resiste nei laboratori di fisici e matematici. O come Delphi, l'ottima ma perdente alternativa di Borland a Visual Basic, la cui undicesima versione è stata rilasciata quest'anno nell'indifferenza generale. O come anche il vetusto C, che pur essendo solo un povero vecchio al cospetto del figlio C++, resta incredibilmente il secondo linguaggio del mondo (dopo Java) nelle classifiche di popolarità.
Nella categoria dei linguaggi non-morti, vecchie glorie siedono al fianco di ambiziosi rampolli che non hanno mai fatto molta strada. Ma non crediate mai di esservene liberati per sempre. Un giorno, qualcuno potrebbe chiamarvi al telefono: "Ho un progetto da affidarti. Si tratta di un vecchio applicativo in Delphi…".

PERCHÉ IMPARARLI
Se non li conoscete già, non ci sono molti motivi per imparare questi linguaggi al giorno d'oggi. Fareste bene a concentrarvi su qualcos'altro. A meno che il vostro datore di lavoro non ve lo chieda, si intende. O non dobbiate modificare qualche vecchio programma. O contribuire a qualche progetto open source che usa il C, primo tra tutti Linux… Insomma, siate sempre pronti ad un attacco di zombi.

PERCHÉ IGNORARLI
La vita è troppo breve, e ci sono cose migliori da studiare.

IL PRIMO LINK
http://en.wikipedia.org/wiki/List_of_programming_languages.

PERL

Perl è una bestia strana almeno quanto il cammello che ne è la mascotte. Fin dagli anni '80, gli utenti tradizionali di questo linguaggio non sono i programmatori, ma piuttosto i sistemisti, che ne hanno fatto largo uso negli anni per produrre script di varia utilità. In particolare, negli anni '90 Perl è diventato uno dei linguaggi più usati per i CGI – quei programmini che siedono sui server dietro le pagine web.
Molti definiscono Perl un linguaggio "di sola scrittura", la cui sintassi rende facile scrivere programmi, ma difficile modificarli. In effetti gli appasionati di Perl promuovo frequenti concorsi di "Perl offuscato" dove si premia il programma più incomprensibile. Ecco il vincitore del 2000, un programma che stampa un orologio in caratteri ASCII:

```
#!/usr/bin/perl
$;="@{'`|;{'^'!.|-
'}";$.++;$.++;$.++;$_="(.)?";/((?{$_.=
$_}).)+$/;@_='~!@#$%^&*(
)_+`-=[]\\{}|;\':",./<>? '=~/$_/;@_
_=$;=~/$_/;$_="(.)*?";/((?{$_.=$_}).)+
$/;$Z-=
$Z;"$.$."-$Z;/((?{$_
_[$z]&&!("${_[$x]}"^"${_[$y]}"^"${_
_[$z]}"^"$Z")&&($a.=$_[$x
],$b.=$_[$y],$z++);$x++;$y+=!($x%="
$.$.");$y%=
"$.$.";}).)+/;$_="^"^"^";$_ _=".>.\
'$_ _ _$b')".".('!\@/\"'^'}.')".']}`';
print;
```

I veri programmi Perl sono un po' più leggibili. Perl è un po' in decadenza, azzoppato dalla sua sintassi marziana, dalla lentezza della sua evoluzione, e dall'estinzione dei CGI in favore di tecnologie più evolute. La macchina virtuale promessa nel 2001 tarda ad arrivare, e il cammello sembra in via di estinzione a favore di pitoni e rubini.

PERCHÉ IMPARARLO
Perché siete degli smanettoni che vogliono sentirsi parte della comunità degli hacker e dei sistemisti Linux.

PERCHÉ IGNORARLO
Linguaggi script più moderni come Python o Ruby sono una scelta migliore in quasi tutti i casi.

IL PRIMO LINK
Un sito apposta per chi comincia: http://learn.perl.org.

GLI SPECIALISTI

La categoria dei linguaggi snob non include solo i linguaggi funzionali, ma anche personaggi più tradizionali che siedono saldamente sul trono della loro piccola nicchia. Lua è un linguaggino di tutto rispetto che è anche molto comune nell'industria dei videogiochi.
Groovy, uno scripting language che ha fatto un po' di rumore in passato ma sembra già in piena decadenza, si è proposto come compagno ideale per Java (e infatti gira sulla stessa macchina virtuale). Smalltalk è il principe dei linguaggi object-oriented, molto poco usato al di fuori dell'università, ma sempre maestro di eleganza e miniera di idee per i puristi della programmazione a oggetti. Probabilmente non vi capiterà di dover imparare questi linguaggi, ma se siete nelle industrie giuste qualcuno potrebbe chiedervelo. E se nessuno ve lo chiede, magari volete impararli per puro piacere intellettuale.

PERCHÉ IMPARARLI
Forse lavorate in un campo dove qualcuno di questi linguaggi è particolarmente popolare. O magari volete scoprire per quale motivo tutto quello che viene inventato oggi è già stato fatto vent'anni fa in Smalltalk. O forse, semplicemente, volete divertirvi un po' con qualcosa di nuovo.

PERCHÉ IGNORARLI
La gran parte delle aziende si tiene alla larga da un programma che quasi nessuno saprebbe modificare.

C++

In principio era il C. I programmatori lo videro, e seppero che era buono. Poi venne la programmazione a oggetti, che prometteva tante cose belle. Nel 1979, un danese con un nome simile ad un incidente d'auto (Bjarne Stroustrup) pensò di accoppiare C e oggetti. Il risultato fu C++, ancora oggi uno tra i linguaggi più popolari e sofisticati.
La filosofia del C++ è la stessa del C: niente è vietato. Niente deve impedire al programmatore di abbandonare ogni prudenza per scrivere un programma velocissimo, o di rompere tutte le barriere di sicurezza e "spaccare il bit" lavorando direttamente in memoria. C++ è un linguaggio da uomini duri, che non temono di spararsi nel piede di tanto in tanto. La compatibilità all'indietro con il C ha permesso a C++ di sedere sul trono dei linguaggi "seri" per anni, fino all'arrivo di Java.
C++ è oggi un linguaggio che invecchia. E' molto più specialistico di qualche anno fa, e non sono molti i masochisti che lo userebbero per sviluppare un sito Web o un'applicazione gestionale. Ma per alcuni settori, non ci sono ancora alternative alla sua potenza bruta e alla sua capacità di lavorare a bassissimo livello. Se dovete scrivere un driver hardware o un sistema operativo, rimboccatevi le maniche: vi toccherà affrontare uno dei linguaggi più complessi del mondo informatico.

PERCHÉ IMPARARLO

Se vi serve, lo saprete. Per molti progetti è ancora indispensabile un linguaggio "a basso livello" e che si compila in codice nativo. Quando questo avviene, C++ è di solito l'unica alternativa sensata. E' ancora uno dei pochi linguaggi che consente un controllo completo di tutto il flusso del codice. Volendolo usare senza nessuna libreria aggiuntiva, macro, o miglioria è virtualmente possibile controllare ogni singolo bit

PERCHÉ IGNORARLO

Dico sul serio. Potreste usarlo per anni senza mai dominarlo a fondo, e le sue potenzialità di produrre bug devastanti ed estremamente insidiosi sono pressoché illimitate.

IL PRIMO LINK

Un buon tutorial su http://www.cplusplus.com

LINGUAGGI STATICI E LINGUAGGI DINAMICI

Esistono linguaggi di ogni tipo: procedurali e funzionali, con e senza oggetti, focalizzati sul web o specializzati nelle applicazioni desktop. Ma la distinzione più importante, quella che ciascun programmatore dovrebbe conoscere, è quella tra linguaggi "staticamente tipati" e linguaggi "dinamicamente tipati", o semplicemente "statici" e "dinamici".

Un linguaggio statico è un linguaggio che ti obbliga a dichiarare il tipo di tutte le variabili. Ad esempio, per creare una stringa in Java si scrive:

```
String s = "Salve, mondo";
```

Ho dichiarato che s è una stringa. Quindi non posso, ad esempio, assegnarle il valore 42. In un linguaggio dinamico, invece, una variabile non ha tipo: assume semplicemente il tipo del valore che contiene. In Ruby, posso scrivere:

```
s = "Salve, mondo"
```

E poco dopo:

```
s = 42
```

Questa differenza non sembra molto importante. Ma le sue conseguenze arrivano molto lontano, tanto che i linguaggi statici e quelli dinamici finiscono per essere fondamentalmente diversi. In un linguaggio statico, il compilatore può controllare che tutti i tipi siano congruenti, eliminando alla radice tutta una serie di possibili bug (questo è il motivo per cui i linguaggi statici sono quasi tutti compilati, e quelli dinamici sono quasi tutti interpretati). Nei linguaggi dinamici (anche chiamati "linguaggi di script"), la mancanza dei tipi permette invece di scrivere codice estremamente flessibile e compatto, abbandonandosi a trucchi e scorciatoie che in un linguaggio statico sarebbero impossibili. Il risultato è che i linguaggi statici come Java, C# o C++ sono ideali per costruire grandi sistemi con grandi team di programmatori, mentre quelli dinamici come Ruby, Python o Perl sono ideali per scrivere brevi programmi, e possono diventare estremamente produttivi in mano a piccoli gruppi di programmatori esperti.

La conclusione? Non imparate un solo linguaggio. Imparatene almeno due, uno statico e uno dinamico. Se conoscete sia Java che Ruby, oppure C# e Python, avrete nel vostro garage tanto l'auto confortevole per i lunghi viaggi che l'agile scooter per rapidi spostamenti in città.

I BRUTTI ANATROCCOLI

Se il nome COBOL non vi fa svegliare la notte urlanti e madidi di sudore, potete ritenervi fortunati. Con le sue oltre 400 parole chiave, questo vetusto linguaggio del 1959 è probabilmente il più brutto tra i linguaggi di programmazione di uso comune. Così brutto da motivare il giudizio del vecchio Dijkstra, uno dei padri dell'informatica: "Il COBOL storpia la mente. Il suo insegnamento dovrebbe quindi essere considerato reato". Volete un esempio? Ecco Hello World in COBOL:

```
IDENTIFICATION DIVISION.
    PROGRAM-ID. HELLO-WORLD.
    PROCEDURE DIVISION.
    PARAGRAPH-1.
        DISPLAY 'Hello, world.'.
        STOP RUN.
```

Se ci mettete dentro qualche espressione matematica, il COBOL sa essere anche più verboso. Ecco un esempio un po' forzato:

```
    MULTIPLY B BY B GIVING B-SQUARED.
    MULTIPLY 4 BY A GIVING FOUR-A.
    MULTIPLY FOUR-A BY C GIVING FOUR-A-C.
    SUBTRACT FOUR-A-C FROM B-SQUARED GIVING RESULT-1.
    COMPUTE RESULT-2 = RESULT-1 ** .5.
    SUBTRACT B FROM RESULT-2 GIVING NUMERATOR.
    MULTIPLY 2 BY A GIVINGDENOMINATOR.
    DIVIDE NUMERATOR BY
DENOMINATOR GIVING X.
```

Nonostante tutto il COBOL è ancora molto usato, soprattutto in campo bancario. Anche il suo cugino ABAP, il linguaggio del sistema gestionale SAP, procura ancora la pagnotta a molti programmatori. Assai diverso, ma nella stessa categoria di linguaggi di cui faremmo volentieri a meno, è il PL/SQL, una terribile estensione proprietaria di Oracle che cerca invano di trasformare il linguaggio di interrogazione SQL in un "vero" linguaggio di programmazione. Vade retro!

PERCHÉ IMPARARLI

Solo perché vi pagano per farlo. Man mano che gli specialisti di questi linguaggi diventano più rari (perché passano ad altri linguaggi, o vanno in pensione), il vostro posto di lavoro potrebbe persino diventare più sicuro per qualche anno. Ma non certo più eccitante.

PERCHÉ IGNORARLI

Sentite quelle piccole urla nella vostra testa? Sono i vostri neuroni che muoiono. Sicurezza o no, forse vi conviene cambiare lavoro.

IL PRIMO LINK

http://www.cobolportal.com.

I NUOVI LINGUAGGI DEL WEB

Se volete vedere il prossimo grande sistema operativo, aprite un browser. Ormai anche le applicazioni web più "impossibili", come i word processor, cominciano a spaventare le tradizionali applicazioni desktop. E quando si tratta di condividere i dati, non c'è gara: il web domina la scena dei network sociali e delle comunicazioni. Probabilmente è solo questione di tempo prima che abbia senso lanciare Photoshop o Excel scrivendo una URL. Ormai è chiaro a tutti che buona parte delle migliori applicazioni che aspettano di nascere nei prossimi anni non avranno bisogno di installazione.
Il problema è che quando si tratta di interfacce grafiche, il browser fa – come dire? - pena. Finalmente, a colpi di JavaScript, siamo riusciti ad avere in un browser funzionalità come il completamento automatico o i correttori ortografici. Ma niente che possa paragonarsi con una sana interfaccia sul desktop, si intende. Per questo motivo, molti pensano che siamo arrivati ad un punto di svolta: il Web 2.0 deve farsi da parte per lasciare spazio alle Rich Internet Application, una serie di nuove tecnologie che permetteranno di avere interfacce moderne senza uscire dal browser. Ovviamente molti ci si stanno buttando. In prima fila c'è ovviamente Microsoft, con il suo sistema Silverlight (che fa girare nel browser programmi in C# e VB.NET). Il candidato ideale, però, è uno dei pochi linguaggi che praticamente tutti i browser del pianeta sono in grado di far girare senza lunghe attese: ActionScript. Adobe si è resa conto di sedere su una miniera d'oro, e ha espanso le vecchie tecnologie di Adobe, come Flash e ActionScript, in una nuova tecnologia di nome Flex. Un tempo Flash serviva solo per scrivere interfacce tamarre e fastidiose "intro" che tutti abbiamo imparato a saltare velocemente. Ora potrebbe diventare il futuro della rete.

PERCHÉ IMPARARLI

A molti fa gola mettere le mani su Internet. Se Adobe e le sue concorrenti hanno visto giusto, tra qualche anno il web sarà inzeppato di "linguaggi da browser" come ActionScript.

PERCHÉ IGNORARLI

La tecnologia dei browser si evolve lentamente. E' logico: nessuno vuole sviluppare un sito che solo il 95% degli utenti sono in grado di visitare. Internet si gioca sulle percentuali, e ci vorrà del tempo prima che una singola tecnologia si imponga sul 100% dei browser.

IL PRIMO LINK

http://www.actionscript.org/resources/categories/Tutorials/.

CONCLUSIONI

Se vuoi imparare un nuovo linguaggio di programmazione, hai solo l'imbarazzo della scelta. Ma qualsiasi linguaggio tu scelga, non cadere nella trappola di usare solo quello per i prossimi vent'anni. Ricorda cosa dice il medico: impara un nuovo linguaggio di tanto in tanto, e il tuo cervello arriverà sempre tonico e scattante all'appuntamento con le prossime tecnologie. Tra le altre cose, molte aziende scelgono di usare una tecnologia piuttosto che un'altra non solo per motivi di stretta utilità ma anche per motivi legati ad aspetti economici e commerciali, per cui farsi trovare pronto ad ogni evenienza è sempre un buon punto di partenza. C'è anche un altro motivo per non fermarsi ad un solo linguaggio: non tutti i linguaggi sono ideali per tutti gli obiettivi. Alcuni dicono che un programmatore completo conosce almeno un linguaggio statico, uno dinamico e un linguaggio di markup. Un obiettivo ambizioso, ma si sa che tutti i grandi viaggi iniziano con un singolo passo: scegli il linguaggio che ti ispira di più, stacca il telefono e accendi il computer. Imparare un linguaggio di programmazione è una splendida avventura, quindi godine ogni minuto. Buon viaggio!

Paolo Perrotta

I PRIMI PASSI

Se non hai mai programmato in vita tua, forse sei intimorito. Non molte persone sono in grado di imparare a programmare. Ma se ne senti la voglia, probabilmente sei già in questa esigua percentuale. Tutto quello che ti serve è un tutorial che inizi davvero da zero.
La gran parte dei tutorial introduttivi per qualsiasi linguaggio non sono, in realtà, del tutto introduttivi. Molti si aspettano che tu conosca già concetti che una persona normale non avrà mai bisogno di apprendere, come quello di "variabile" o di "funzione". Un ottimo tutorial per cominciare da zero con il linguaggio Ruby è "Learning to Program", di Chris Pine. E' un libro, ma ne esiste una versione ridotta gratuita sul web all'indirizzo http://pine.fm/LearnToProgram. Purtroppo è in inglese, ma state tranquilli: è tradotto anche in Francese e Giapponese. Come dite? Preferite la buona vecchia lingua di Dante? Mi sembra giusto. Le risorse in italiano sono più ridotte, ma esistono. Oltre a questa rivista, una buona risorsa in italiano è "Pensare da Informatico", su http://www.python.it/doc/Howtothink/Howtothink-html-it/index.htm. Anche questa è la traduzione di un libro: "How to Think Like a Computer Scientist" di Downey, Elkner e Meyers, che usa Python per introdurre i concetti della programmazione. L'originale (che pure è gratuito) è in inglese, ma il link ti porterà alla traduzione italiana di Alessandro Pocaterra. Buona lettura!

I FUNZIONALI

I più snob tra i linguaggi di programmazione sono i cosiddetti linguaggi funzionali. Il loro storico capostipite Lisp, nato nel 1958, è ancora considerato da molti il linguaggio concettualmente più elegante mai concepito. L'abbondanza di parentesi tonde, però, rende la sua sintassi poco appetibile ai gusti delle masse. Ecco un programma per il calcolo del fattoriale scritto in Lisp:

```
(defun factorial (n)
  (if (<= n 1)
    1
    (* n (factorial (- n 1)))))
```

Un membro più moderno della categoria dei linguaggi funzionali è il linguaggio Haskell, del 1990, che a volte viene selezionato per scrivere sistemi particolarmente critici. Il ramo più giovane della famiglia è quello dei cosidetti "linguaggi concorrenti", specializzati nello sviluppo di sistemi paralleli. Il più alla moda di questi è attualmente il linguaggio Erlang, sviluppato inizialmente da Eriksson per costruire grandi sistemi di telefonia.
I linguaggi funzionali sono animali bizzarri e affascinanti. Hanno ben poco in comune con i linguaggi procedurali e con i linguaggi a oggetti che imperano nel mondo della programmazione: se decidete di impararli, preparatevi ad una serie di shoccanti rivelazioni. Ad esempio, Erlang non ha il concetto di variabile!

PERCHÉ IMPARARLI

Prima di tutto perché niente espande la mente come un modo completamente diverso di fare le cose. Ma anche perché i linguaggi funzionali stanno attraversando un periodo di riscatto. Linguaggi come Erlang sono più adatti dei linguaggi tradizionali a problemi oggi comuni, come gestire enormi database su Internet, o scrivere software parallelo per i processori multi-core.

PERCHÉ IGNORARLI

La scienza non sempre paga la pagnotta, e non sempre la soluzione tecnicamente migliore è quella che alla fine si impone sul mercato. I linguaggi funzionali potrebbero restare una curiosità da snob per altri cinquant'anni.

IL PRIMO LINK

http://www.erlang.org/download/getting_started-5.4.pdf.

A Microsoft, questa faccenda di Java non è mai andata giù. Ma come? Tutta questa fatica per dominare il mercato con Windows, e improvvisamente arriva un linguaggio che tratta il sistema operativo con spocchiosa indifferenza? E' per questo che nel 2001, l'aziendona di Zio Bill pubblica la sua alternativa a Java: un linguaggio quasi identico a Java, con una Virtual Machine come Java (parte di un sistema che Microsoft ha chiamato .NET), ma inizialmente solo per Windows. Che C# sia un clone di Java non c'è dubbio, ma a volte l'allievo supera il maestro. C# non esita a sperimentare funzionalità nuove; e mentre Java si preoccupa della propria fama di linguaggio "serio", C# non si vergogna di semplificare in campi essenziali come le interfacce grafiche e i database. Alla fine è spesso Java che deve adeguarsi, seguendo con un po' di affanno la via tracciata dal rivale. E grazie a progetti open source come Mono è oggi possibile, con qualche sacrificio, far girare C# su sistemi diversi da Windows.

PERCHÉ IMPARARLO

In generale è un po' più produttivo di Java, almeno per i piccoli progetti. E magari la vostra azienda preferisce roba di Microsoft, o volete ritagliarvi una nicchia in un mercato saturo di javisti.

PERCHÉ IGNORARLO

È come Java, ma senza il dinamico mondo open source che circonda Java. La gran parte dei progetti interessanti nel mondo C# sono conversioni di progetti Java. E quando si tratta di compatibilità multi-piattaforma, Java resta inarrivabile.

IL PRIMO LINK

La guida ufficiale di Microsoft su http://msdn2.microsoft.com/en-us/vcsharp.

RUBY

Ruby è giovanile, ma ha quasi quindici anni. Fino al 2000 è rimasto confinato al Giappone, suo paese di origine. Poi qualcuno in occidente ha scoperto questo linguaggio di script potente, più "orientato agli oggetti" di Python e più elegante di Perl, e Ruby si è ritagliato la sua nicchia di fan. Ma il meglio doveva ancora arrivare.
Nel 2004, un programmatore danese ha pubblicato Ruby on Rails, un framework basato su Ruby per sviluppare siti web. E' stato un terremoto, con seri professionisti che testimoniavano incrementi di produttività fantascientifici, e le comunità di tutti i linguaggi che si affannavano a sviluppare nuovi framework sull'esempio di Rails. Per citare Nathan Torkington, un esperto di open source: "Guardare Ruby on Rails è come guardare uno di quei film di kung-fu dove una dozzina di energumeni si apprestano a malmenare il ragazzino appena arrivato in città, e vengono riempiti di ceffoni in una gran varietà di modi pittoreschi".
E così Ruby, come quegli attori che restano inchiodati al loro ruolo di maggiore successo, è diventato famoso come "il linguaggio di Rails". Ma i suoi appassionati giurano che è solo questione di tempo prima che Ruby conquisti il mondo, o almeno un paio di continenti.

PERCHÉ IMPARARLO

In una parola: Rails. Ma non solo. La comunità Ruby è attualmente il quartiere più fico della città, quello dove gli artisti e i personaggi famosi tirano fuori idee nuove. E Ruby è anche un ottimo linguaggio didattico, ideale per chi impara a programmare per la prima volta.

PERCHÉ IGNORARLO

Qualcuno dice che è solo una moda passeggera. Se preferite un valido concorrente come Python, potete anche aspettare e scoprire se la Ruby-mania passa da sé.

IL PRIMO LINK

Try Ruby, un fantastico minicorso interattivo online: http://tryruby.hobix.com.

I MARKUP LANGUAGES

Ma HTML, sarà un linguaggio? Sì, ma non certo un linguaggio di programmazione. Lo stesso vale per XML, l'altro onnipresente linguaggio di markup. XML, però, è usato spesso come base per "veri" linguaggi, ancorché di dubbia leggibilità. Un esempio particolarmente sgraziato è XSLT, per trasformare dati XML che è a sua volta basato su XML. Ecco Hello World in XSLT:

```
<xsl:stylesheet version="1.0"
 xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
   <xsl:template match="/">
      <xsl:text>Hello World!</xsl:text>
   </xsl:template>
</xsl:stylesheet>
```

Bleah! Ma XSLT è molto utile quando dovete trasformare un file XML, quindi nonostante la sua brutta sintassi potrebbe toccarvi di usarlo. In generale, i markup language sono una buona idea quando si tratta di rappresentare dati. E oggi ci sono alternative più eleganti (anche se meno popolari) ad XML, come YAML o JSON, che si imparano in un attimo e si integrano con tutti i linguaggi più comuni.

PERCHÉ IMPARARLI

Che vi piacciano o meno, HTML, XML e i loro innumerevoli dialetti sono uno strumento essenziale per moltissimi programmatori.

PERCHÉ IGNORARLI

Fatelo solo a vostro rischio.

IL PRIMO LINK

http://www.w3schools.com.

PYTHON

Creato nel 1991 dal "benevolo dittatore" Guido van Rossum, Python (il cui nome non si ispira ai rettili, ma ai comici inglesi Monty Python) ha cambiato le carte in tavola. Prima di Python esistevano i linguaggi "seri", quelli per costruire sistemi, e i "linguaggi di script", buoni al massimo per scrivere semplici programmini di utilità. Python ha dimostrato negli anni che un linguaggio interpretato può essere non solo più produttivo di un Java o di un C++, ma anche più elegante ed espressivo. Python è in un certo senso un linguaggio di nicchia. Le sue nicchie, però, sono talmente tante da renderlo uno dei linguaggi più amati e popolari. Dagli effetti speciali per il cinema ai siti web, dall'industria dell'elettronica all'amministrazione di sistema, Python è un linguaggio che conta eserciti di fan sfegatati. E se un giorno tutto il codice Python dovesse sparire, lo sapremmo tutti subito: il primo ad andare giù sarebbe Google, che lo usa in abbondanza.

PERCHÉ IMPARARLO

Un bel linguaggio di script moderno, utile per i vostri lavoretti quotidiani ma anche per scrivere programmi "seri". Alcuni framework, come Django, meritano attenzione. Ed è sicuramente più veloce di Ruby, il suo più vicino concorrente.

PERCHÉ IGNORARLO

Il cugino Ruby, grazie anche al suo modello object-oriented più "puro", sta accentrando tutta l'attenzione su di sè. E per quanto Python sia sicuramente un linguaggio utile per chiunque, non sono molte le aziende che vi assumeranno perché avete aggiunto un pitone al vostro curriculum. A meno che non stiate parlando di Google che, notoriamente fa largo uso di Python

IL PRIMO LINK

Il libro Dive Into Python, su http://www.diveintopython.org.

NOTA

OGGETTI O NON OGGETTI

Quando sceglie l'automobile, un bravo acquirente chiede al concessionario: è sicura? Allo stesso modo, un bravo programmatore che vuole scegliere un nuovo linguaggio dovrebbe sempre chiedere: è orientato agli oggetti? Se non sapete cos'è la programmazione a oggetti, state tranquilli. Studiando il vostro prossimo linguaggio probabilmente lo imparerete. Banalizzando, possiamo dire che tanto più un linguaggio è "orientato agli oggetti", tanto più è moderno e potente.

I trucchi del mestiere

Tips & Tricks

Questa rubrica raccoglie trucchi e piccoli pezzi di codice, frutto dell'esperienza di chi programma, che solitamente non trovano posto nei manuali. Alcuni di essi sono proposti dalla redazione, altri provengono da una ricerca su Internet, altri ancora ci giungono dai lettori. Chi volesse contribuire, potrà inviare i suoi Tips&Tricks preferiti. Una volta selezionati, saranno pubblicati nella rubrica. Il codice completo dei tips è presente nel CD allegato nella directory *tips*\\ o sul Web all'indirizzo: cdrom.ioprogrammo.it.

VB.NET

COME POSSO APRIRE UN PDF?

```
Public Class Form1

    Private Sub Button1_Click(ByVal sender As System.Object,
                ByVal e As System.EventArgs) Handles Button1.Click

        Dim proc As New Process()

        With proc.StartInfo
            .Arguments = "Your PDF Path eg:-
                                C:\MyFiles\Ebook2007.pdf"
            .UseShellExecute = True
            .WindowStyle = ProcessWindowStyle.Maximized
            .WorkingDirectory = "C:\Program Files\Adobe\Reader
                        8.0\Reader\" '<----- Set Acrobat Install Path
            .FileName = "AcroRd32.exe" '<----- Set Acrobat Exe
                                                        Name
        End With

        proc.Start()
        proc.Close()
        proc.Dispose()

    End Sub

End Class
```

COME POSSO GENERARE UNA PASSWORD CASUALE?

```
Dim rNum As New Random(100)
    Dim rLowerCase As New Random(500)
    Dim rUpperCase As New Random(50)
    Dim psw As String
    Dim RandomSelect As New Random(50)

    Public Function Gen_Psw(ByVal Lenght As Integer, Optional
                    ByVal Reset As Boolean = False) As String
        Dim i As Integer
        Dim CNT(2) As Integer
        Dim Char_Sel(2) As String
        Dim iSel As Integer
        'clear old passwords
        If Reset = True Then
            psw = ""
        End If
        For i = 1 To Lenght
            'create random numbers that will represent
            'each : upercase,lowercase,numbers

            CNT(0) = rNum.Next(48, 57) 'Numbers  1 to 9
            CNT(1) = rLowerCase.Next(65, 90) ' Lowercase
                                                Characters
            CNT(2) = rUpperCase.Next(97, 122) ' Uppercase
                                                Characters
            'put characters in strings
            Char_Sel(0) = System.Convert.ToChar(CNT(0)).ToString
            Char_Sel(1) = System.Convert.ToChar(CNT(1)).ToString
            Char_Sel(2) = System.Convert.ToChar(CNT(2)).ToString

            'pick one of the three above for a character At Random
            iSel = RandomSelect.Next(0, 3)
            'colect all characters generated through the loop
            psw &= Char_Sel(iSel)

            ' reset  with new password
            If Reset = True Then

                psw.Replace(psw, Char_Sel(iSel))
            End If

        Next
        Return psw

    End Function
```

COME POSSO CONOSCERE LA DIMENSIONE DI UNA CARTELLA?

```
Imports Microsoft
Imports Microsoft.Win32
```

```
Imports Microsoft.Win32.Registry
Imports System.Collections
Imports System.Windows.Forms
Imports System.IO
Imports System.Threading

' Returns the sum of the files in the folder.
  ' dPath: Path of the directory
  ' include subfolders: set if include subfolders ;)
  Public abort As Boolean
  Function GetFolderSize(ByVal DirPath As String, ByVal
                    includeSubFolders As Boolean) As Long
    Try
      Dim size As Long = 0
      Dim diBase As New DirectoryInfo(DirPath)
      Dim files() As FileInfo
      If includeSubFolders Then
        files = diBase.GetFiles("*",
                        SearchOption.AllDirectories)

      Else
        files = diBase.GetFiles("*",
                        SearchOption.TopDirectoryOnly)
      End If
      Dim ie As IEnumerator = files.GetEnumerator
      While ie.MoveNext And Not abort

        size += DirectCast(ie.Current, FileInfo).Length
      End While
      Return size
    Catch ex As Exception
      MsgBox("Error: " & ex.Message)
      Return -1
    End Try
  End Function      }
```

C#

COME POSSO STAMPARE UNA SERIE DI FIBONACCI?

```
static int Fibonacci (int x)
{
   Console.WriteLine ("x = {0}", x);
   if (x <= 1)
   {
      return 1;
   }
   return Fibonacci (x-1) + Fibonacci (x-2);
}

static void Main( )
{
   Console.WriteLine ("Fibonacci no. = {0}", Fibonacci (5));
   Console.ReadKey();
}
```

COME POSSO CONVERTIRA UNA STRINGA IN UN NUMERO?

```
try
{
   int number = Convert.ToInt32 (Console.ReadLine());
   Console.WriteLine ("Number: " + number);
}
catch (FormatException e)
{
   Console.WriteLine (e.Message);
}
 // or ...

try
{
   int number = Int32.Parse (Console.ReadLine());
   Console.WriteLine ("Number: " + number);
}
catch (ArgumentNullException e)
{
   Console.WriteLine (e.Message);
}
Come posso effettuare l'override del metodo tostring()?
public class MyClass
{
   private string customer ="";
   private int customerID = 0;
    public string Customer
   {
      get { return customer; }
      set { customer = value; }
   }
    public int CustomerID
   {
      get { return customerID; }
      set { customerID = value; }
   }
    public override string ToString()
   {
      return string.Format ("Customer = {0} ID = {1}", Customer,
                                        CustomerID);
   }
 }
    End Sub
```

COME POSSO CONVERTIRE UN COLORE DA RGB IN HEX?

```
Private Sub ConvertHtmlToRTF()
' I sample to read from html file
Dim sHtml As String =
                    IO.File.ReadAllText("C:\TestHtml.htm")

' I will save it to temp folder
If Dir("C:\Temp", FileAttribute.Directory) = "" Then
System.IO.Directory.CreateDirectory("C:\Temp")
End If

' Create rtf File
Dim b As System.IO.TextWriter = New
    System.IO.StreamWriter("C:\Temp\test.rtf", False,
                          System.Text.Encoding.Unicode)
' Write it to rtf format
b.Write(sHtml)

' Clear object
b.Flush()
b.Close()
' Open file
System.Diagnostics.Process.Start("C:\Temp\test.rtf")

End Sub
```

COME POSSO CREARE UN'ANIMAZIONE?

```
mport java.applet.Applet;
import java.awt.Graphics;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;

public class AppletAnimation extends Applet
                              implements Runnable {
  int frameNumber = -1;

  int delay = 100;

  Thread animatorThread;

  boolean frozen = false;

  public void init() {
    String str;
    addMouseListener(new MouseAdapter() {
      public void mousePressed(MouseEvent e) {
        if (frozen) {
          frozen = false;
          start();
        } else {
          frozen = true;
          stop();
        }
      }
    });
  }
  public void start() {
    if (!frozen) {
      if (animatorThread == null) {
        animatorThread = new Thread(this);
      }
      animatorThread.start();
    }
  }
  public void stop() {
    animatorThread = null;
  }
  public void run() {
Thread.currentThread().setPriority(Thread.MIN_PRIO
                                             RITY);
    long startTime = System.currentTimeMillis();

    Thread currentThread = Thread.currentThread();
    while (currentThread == animatorThread) {
      frameNumber++;
      repaint();
      try {
        startTime += delay;
        Thread.sleep(100);
      } catch (InterruptedException e) {
        break;
      }
    }
  }

  public void paint(Graphics g) {
    g.drawString("Frame " + frameNumber, 0, 30);
  }
}
```

COME POSSO SAPERE SE UNA PAGINA È STATA CAMBIATA DI RECENTE?

```
<?php
$latesttime = 0;
if ($handle = opendir('/home/shouri/public_html')) {
    while (false !== ($file = readdir($handle))) {
        $recenttime = filemtime($file);
        if ($recenttime > $latesttime )
            $latesttime = $recenttime;
    }
    closedir($handle);
    echo("Last updated on " . date("F d Y",
        $latesttime) . " at " . date("H:i", $latesttime) . "
                                                    IST");
}
?>
```

MACROMEDIA FLEX: FLASH È SERVITO

NELL'AFFRONTARE LO SVILUPPO DI UN'APPLICAZIONE WEB-ORIENTED ABBIAMO UNA VASTA SCELTA DI STRUMENTI A CUI RIVOLGERCI. TRA QUESTI, ADOBE FLEX 2 SI RIVELA PARTICOLARMENTE INTERESSANTE. VEDIAMO PERCHÈ...

Esistono due punti di vista, quello del designer e quello dello sviluppatore, spesso agli antipodi. L'ambiente di Flash MX è adatto ad un designer. Tipicamente uno sviluppatore non è abituato a pensare in termini di grafica e timeline, questi sono concetti cari a chi fa animazione o a chi utilizza la grafica vettoriale. Tipicamente un programmatore è abituato a pensare in termini di classi, oggetti, eventi e algoritmi. In Adobe hanno deciso di fornire a chi sviluppa uno strumento la cui finalità è la stessa di macromedia flash, ovvero creare delle applicazioni ricche di grafica ed effetti multimediali, ma raggiungendo lo scopo con un approccio più programmativo e meno visuale.In questo articolo realizzeremo una completa applicazione Flex e inizieremo a comprendere la logica di questo framework.

DI COSA STIAMO PARLANDO?

Vediamo allora innanzitutto come installare, configurare e muovere i primi passi con Flex. Per l'applicazione di esempio useremo l'ambiente Flex Builder rilasciato da Adobe. Questo è un ambiente commerciale, potete trovarlo allegato al numero di Agosto (117) di ioProgrammo, o se preferite potete scaricarlo dal sito di Adobe (per l'url vedi il box laterale). A differenza del Flex Builder, il Flex SDK, che è tutto ciò di cui abbiamo realmente bisogno per sviluppare è Open Source. Avendo a disposizione il Flex SDK è possibile quindi scrivere il codice del nostro progetto con l'editor che più ci è congeniale e compilarlo tramite il compilatore a riga di comando presente nell'SDK stesso.
E' chiaro come l'ambiente di sviluppo integrato sia maggiormente user-friendly dandoci la possibilità di costruire l'interfaccia dell'applicazione in maniera visuale e scrivere il codice con l'aiuto del syntax highlighting, ma anche con i tools a riga di comando possiamo svolgere egregiamente il nostro lavoro.

IL FLEX BUILDER

Una volta installato il Flex Builder (da ora FB), diamo un'occhiata alle sue caratteristiche. Come ogni ambiente di sviluppo, il Flex Builder mette a disposizione dell'utente una serie di viste sul codice che si sta scrivendo. Il termine usato per indicare queste viste è *perspective*. Ad esempio l'editor mediante il quale digiteremo il codice della nostra applicazione viene indicato come *Development Perspective Source Mode* mentre quando lanciamo il debug, l'ambiente passa in quella che viene definita come *Debugging Perspective*.
Un'altra vista per noi comoda ed interessante è la *Development Perspective Design Mode*, cioè quella che ci permette di comporre l'interfaccia dell'applicazione in maniera visuale. Bene, vediamo ora dal vivo il significato di queste viste.

PRIMO CONTATTO

Dopo aver fatto partire FB, andiamo nel menu File | New e scegliamo la voce Flex Project.

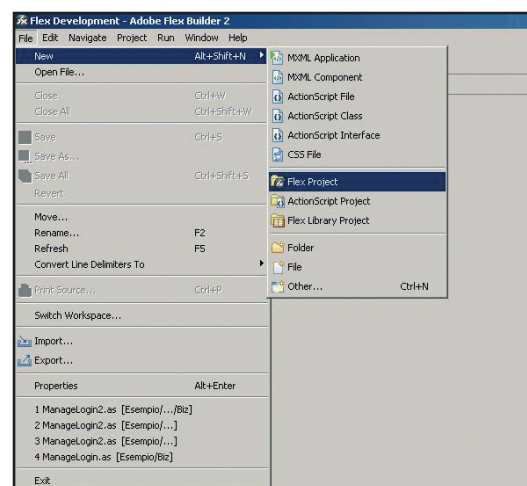



Figura 1: Scegliamo il progetto Flex.

Come vi sarete accorti, il menu File | New, oltre ad un progetto di tipo Flex, permette anche altri tipi di

progetti e cioè, Flex Library Project, cioè una libreria che sarà compilata in un file .SWC e che successivamente potremo aggiungere ai nostri progetti, oppure ActionScript Project, per scrivere puro codice actionscript.
Ma non divaghiamo. Dopo aver scelto il progetto di tipo Flex, il wizard ci chiede di indicare se e come accedere ai dati (intendendo la parola dati in maniera molto generica). Scegliamo la voce Basic.
A questo punto dobbiamo indicare il nome del nostro progetto e la directory di lavoro. Dopo aver inserito anche queste ultime informazioni, il progetto viene creato e sarà visibile nella vista Navigator. L'ambiente ci chiede ancora alcune informazioni per concludere la configurazione del progetto. Queste informazioni non sono assolutamente necessarie in questo momento e potrebbero essere specificate anche successivamente. Infatti riguardano cartelle che contengono codice sorgente e che si trovano al di fuori della cartella principale del progetto, o eventuali librerie esterne da usare nel progetto.
Per il momento non vogliamo indicare nessuna di queste informazioni, quindi premendo il pulsante *Finish* del wizard possiamo completare la creazione del progetto.
Al termine della creazione, l'FB si sposta nella vista *Source Mode* ed all'interno dell'editor, possiamo vedere il codice fondamentale della nostra applicazione. Il codice è di estrazione XML, ed inoltre possiamo notare che l'estensione del file appena creato è .mxml.
L'MXML rappresenta il metalinguaggio con cui possiamo realizzare la nostra interfaccia. Questo linguaggio, come detto prima, è un dialetto XML ed i suoi tag permettono di definire i vari aspetti grafici che la nostra applicazione deve presentare all'utente. Il codice che sovraintende all'interazione con l'interfaccia grafica, invece, verrà scritto in Actioscript e potrà essere presente nello stesso file che contiene i tag MXML oppure in uno separato.
Ma vediamo di capire meglio la cosa con un esempio.
L'applicazione che abbiamo appena realizzato è costituita dal seguente codice MXML:

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application
      xmlns:mx="http://www.adobe.com/2006/mxml"
layout="absolute">

</mx:Application>
```

La prima riga è la classica intestazione di un file XML. Il tag *mx:Application* rappresenta il contenitore grafico principale di una generica applicazione Flex. In questo tag possiamo specificare una serie di informazioni che avranno validità per l'intera applicazione. Le due cose che il wizard di creazione imposta per noi sono il *namespace* ed il *layout* di default. Tutti i controlli disponibili in Flex sono infatti presenti nel namespace mx e la sua dichiarazione nel tag Application rende più semplice accedervi. Il layout invece viene impostato ad "absolute" che vuol dire che siamo liberi di posizionare dove vogliamo i controlli all'interno dell'area dell'applicazione. Se infatti scegliessimo una delle altre opzioni, "horizontal" o "vertical", tutto quello che trasciniamo nell'area dell'applicazione di disporrà automaticamente in base al tipo di layout scelto, cioè orizzontalmente o verticalmente. La proprietà Layout, ed in genere tutte le proprietà di un controllo, possono essere modificate mediante la finestra delle proprietà che è visibile in *Design View*.

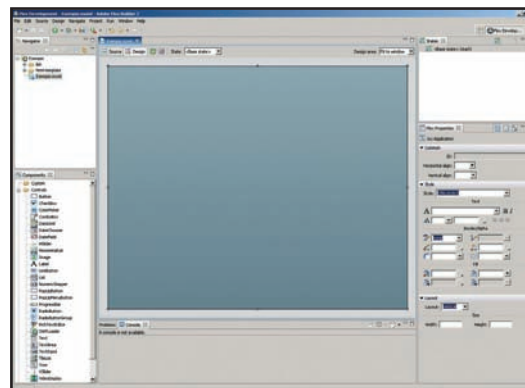



Figura 2: L'ambiente Flex Builder 2

La figura 2 mostra l'ambiente del FB appunto in modalità design, e si possono vedere, a sinistra l'elenco dei controlli, mentre a destra la finestra di riepilogo delle proprietà.
Al centro vediamo invece come si presenta il tag Application in modalità design, e cioè come un'area su cui possiamo andare a disporre i controlli che comporranno l'interfaccia della nostra applicazione.

I CONTROLLI

Come potete vedere nella finestra *Components*, i controlli sono suddivisi in categorie:

- **Custom**: Sono quelli realizzati da noi o da terze parti.
- **Controls**: Sono i controlli standard che permettono l'interazione con l'utente e che hanno una interfaccia grafica, tipo Bottoni, CheckBox, ComboBox, DataGrid, etc
- **Layout**: Permettono di definire l'aspetto dell'applicazione o di sottoaree del layout dell'ap-

NOTA

Il Flex Builder ed il Flex SDK possono essere scaricati dal sito della Adobe all'indirizzo: http://www.adobe.com/products/flex/. Chiaramente, l'installazione del Flex Builder include quella del Flex SDK.

plicazione.

- **Navigators**: Qui sono raccolti controlli di tipo navigazionale, quali l'Accordion, MenuBar, TabBar, etc
- **Charts**: Raccoglie varie tipologie di grafici.

Per la nostra applicazione noi lavoreremo prevalentemente con controlli presenti nelle categorie Controls, Layout e Navigators ma il modo in cui li utilizzeremo vale anche per quelli delle altre categorie.

NOTA

Per cominciare subito a lavorare, copiate il codice allegato all'articolo sul vostro HD. Poi aprite il Flex Builder, quindi dal menu File | New, scegliete Import e poi la voce Existing Projects into Workspace. Indicate la cartella che contiene il file .mxml principale del progetto, ed il gioco è fatto.

DEFINIAMO L'INTERFACCIA UTENTE

Cominciamo quindi a rendere presentabile l'applicazione. Per prima cosa dobbiamo definire una maschera per il login. Infatti, la nostra applicazione MusicShop avrà due interfacce, una che permetterà ai clienti di fruire delle informazioni presenti, ed una con cui l'amministratore potrà organizzare queste informazioni. Nel corso di questa serie di articoli l'attenzione sarà particolarmente rivolta allo sviluppo tramite Flex ed Actionscript. Di conseguenza il codice della parte server sarà esplicitato la dove necessario al chiarimento di ciò che si sta realizzando. Lo sviluppo o l'estensione del codice lato server è lasciato a voi lettori come esercizio.
Per cominciare quindi costruiamo la nostra maschera di login. Spostiamoci in modalità design e visualizziamo i controlli di tipo Layout. Indiviamo il controllo *Panel* e trasciniamolo sull'area del controllo *Application*.
Una volta fatto ciò, passando in modalità *development* potrete vedere questo codice nell'editor:

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application
     xmlns:mx="http://www.adobe.com/2006/mxml"
                           layout="absolute">
          <mx:Panel width="250" height="200"
                                layout="absolute"
x="275" y="24">
          </mx:Panel>
</mx:Application>
```

Ora possiamo renderci conto di come il tag Application sia realmente il contenitore generale dell'applicazione. Tutti gli altri controlli saranno contenuti in esso, e se sono presenti dei controlli di tipo layout, a loro volta questi conterranno altri controlli e così via.
Un'altra cosa di cui possiamo renderci conto è di come sia facile scrivere il codice MXML anche con un normale editor, chiaramente conoscendone e rispettandone la sintassi.

Torniamo in modalità *design*. Selezioniamo il controllo Panel appena inserito cliccandoci sopra, e modifichiamo alcune sue proprietà. Inseriamo l'ID che servirà ad identificare il controllo quando vorremo riferirci ad esso via codice, il Titolo, ed impostiamo i valori per l'anchor mediante la voce *Constraints* visibile nella parte bassa della finestra delle proprietà. Selezioniamo i checkbox centrali, sia in alto che a destra. Questo significa che abbiamo ancorato il Panel in posizione centrale rispetto al suo controllo padre, cioè il tag Application.
A questo punto, inseriamo due controlli *Label*, due controlli *Inputbox* e due controlli *Button* in modo da ottenere una cosa simile alla Figura 3.

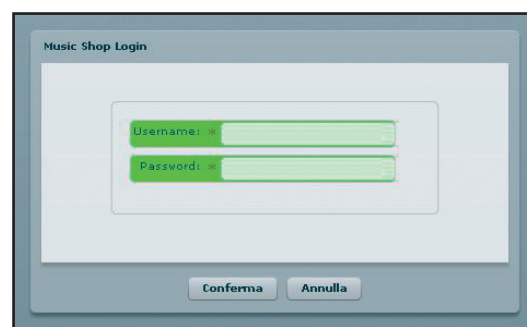



Figura 3: La maschera di Login del Music Shop

Come vedete, la finestra delle proprietà permette di impostare solo alcune delle proprietà di un controllo. Per poterle visualizzare tutte, dovete scegliere la vista alfabetica o quella per categoria mediante i pulsanti posti, sempre nella finestra delle proprietà, in alto a destra.
Tra quelle immediatamente modificabili mediante la finestra delle proprietà ce ne sono un paio che attirano la nostra attenzione. La prima si chiama *Corner Radius* ed ha per simbolo un angolino, la seconda *Alpha*, posta lì nei pressi. La proprietà corner radius permette di arrontondare gli angoli di tutti i controlli grafici che espongono questa proprietà. Alpha invece agisce sulla trasparenza del controllo. Impostando quindi queste due proprietà è possibile rendere più scenografica la nostra interfaccia. Ad esempio, selezioniamo il controllo Panel ed i due controlli InputText e impostiamo la proprietà corner radius a 5.
Poi selezioniamo il Panel ed impostiamo la proprietà Alpha a 0.80.
Quindi aggiungiamo un controllo *Image*. Posizioniamolo in modo che ricopra il Panel e impostiamo la sua proprietà source, specificando l'immagine da visualizzare. Vi sarete accorti che l'immagine nasconde il Panel, dato che questo si trova sotto. Purtroppo non c'è la possibilità da menu di definire lo z-order dei controlli. Possiamo farlo comunque in maniera molto semplice dalla *Source View*

selezionando il tag relativo all'immagine e spostandolo prima di quello relativo al Panel. Tornando in *Design View*, noterete le differenza. Dal menu *Run* selezioniamo la voce *Run* seguita dal nome del file principale dell'applicazione. In questo modo lanciamo la compilazione ed esecuzione della nostra applicazione che sarà caricata in una istanza del browser di default.
Il risultato sarà quello indicato in Figura 4.

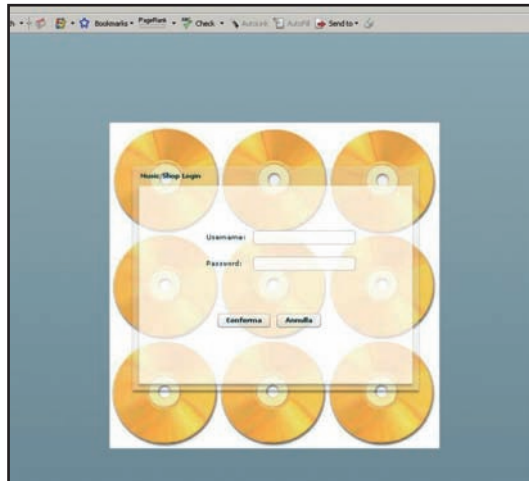



***Figura 4:** La maschera di Login del Music Shop*

INTERAZIONE CON L'UTENTE

Bene a questo punto abbiamo una interfaccia utente, ma per renderla fruibile dobbiamo far sì che quando premiamo i pulsanti, qualcosa accada. A questo proposito definiamo gli event handlers dei pulsanti. In particolare dobbiamo gestire l'evento *click*, che viene scatenato alla pressione del tasto.
Passiamo in modalità design e selezionamo il pulsante "Conferma". Notiamo che tra le proprietà del pulsante è presente una indicata come *OnClick*. Questa permette di indicare il nome di una funzione che verrà richiamata quando premeremo il pulsante e che quindi fungerà da event handler. Scriviamo "OnConfirm()".
Se eseguiamo l'applicazione, noteremo la presenza di un errore, segnalato nella view *Problems*. Infatti noi abbiamo detto che l'event handlers del pulsante "Conferma" è la funzione di nome OnConfirm, ma in effetti questa...non esiste !!
Vediamo come aggiungere il nostro primo pezzo di codice ActionScript.
Come ho detto precedentemente è possibile inserire il codice ActionScript all'interno di un file .mxml insieme ai vari tags, oppure in un file separato. Se scegliamo di inserire il codice del nostro event handler nello stesso file .mxml dove si trova il pulsante, passiamo in *Source View* e spostiamo il cursore sulla riga immediatamente precedente il tag di chiusura </application>. Qui digitiamo il tag <mx:Script>. Appena inserita la parentesi ">" di chiusura, l' FB completa il tag script in questo modo:

```
<mx:Script>
    <![CDATA[

    ]]>
</mx:Script>
```

All'interno del tag CDATA sarà possibile inserire le istruzioni ActionScript, solitamente organizzate in funzioni. Quindi per definire il nostro event handler scriviamo così:

```
<mx:Script>
    <![CDATA[
        import
            mx.controls.Alert;

        private function
            OnConfirm():void
        {
mx.controls.Alert.show("Hai premuto Conferma !!",
"Music Shop");
        }
    ]]>
</mx:Script>
```

OnConfirm è una funzione privata, quindi utilizzabile solo all'interno del file dove è dichiarata e non ritorna informazioni, questo è indicato dal termine *void* posto subito dopo i ":". Infatti se una funzione ritorna una qualche informazioni al suo chiamante, il tipo dell'informazione ritornata va indicato dopo le parentesi in questo modo:

private function NomeFunction() : *tipoInformazioneRitornata*

Ritornando al nostro esempio, quando viene premuto il pulsante "Conferma", la OnConfirm si limita a richiamare una funzione presente nella libreria di sistema *mx.controls.Alert* per visualizzare un messaggio all'utente.
Bello, ma comunque poco funzionale. Nella realtà dovremmo controllare che sia lo username che la password vengano inserite. Tralasciando questo piccolo dettaglio, vediamo come fare se vogliamo organizzare il nostro codice in modo più ordinato. Infatti, pur essendo liberi di scrivere il codice come abbiamo appena visto, se l'applicazione richiede molti controlli e molto codice ActionScript per governare le sue funzionalità, rischieremmo di avere presto un file, molto denso

ed alla fine poco gestibile e manutenibile.
Prima ho accennato al fatto che è possibile scrivere il codice ActionScript in un file separato. Questa opportunità può facilitarci la vita, e non solo nel caso di applicazioni complesse.

CODICE ACTIONSCRIPT IN FILES ESTERNI

Ci sono due modi per scrivere codice in un file file esterno.
Nel primo, il file si trova comunque all'interno della directory di lavoro principale della nostra applicazione. Per creare un file ActionScript, dal menu File | New scegliamo la voce *ActionScript Class*. FB presenterà una finestra, Figura 5, mediante la quale inserire le informazioni che ca-

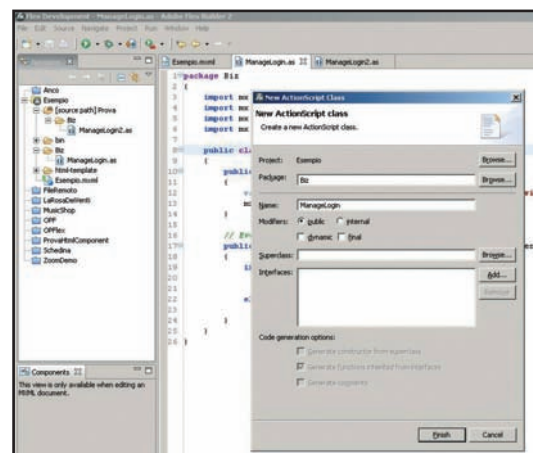



Figura 5: jog

ratterizzano la classe. In questo modo creeremo un file di nome *ManageLogin.as* che conterrà la classe ManageLogin del package Biz.
Come potete notare file e classe hanno lo stesso nome. Inoltre package e directory che contiene il file hanno lo stesso nome. Queste corrispondenze non sono casuali. I packages, sono dei contenitori logici che possono raccogliere una o più classi. La loro controparti sull'Hard Disk sono le cartelle. Le classi in un package, corrispondono ai files presenti nella cartella con il nome del package. In questo modo si ottiene una organizzazione sia logica che fisica del nostro codice. Ora supponiamo che il file esterno sia già stato scritto, magari per un'altra applicazione, e si trovi in una cartella al di fuori di quella principale del nostro progetto. Per poterlo utilizzare dovremo indicare al compilatore quale è il nome di questa cartella "esterna". Quindi, selezioniamo il progetto nella vista *navigator*, richiamiamo il menu contestuale con il tasto destro del mouse, scegliamo *properties* e quindi *Flex Build Path*. In questo pannello, permiamo il tasto *Add Folder* e aggiungiamo la cartella in cui si trova il file.
Il tutto è riepilogato dalla Figura 6. Da questo

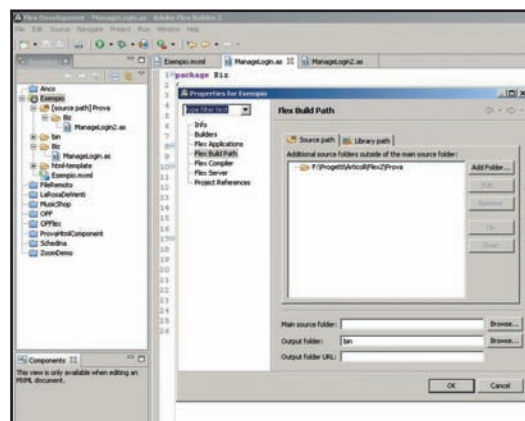



Figura 6: Pannello di configurazione delle proprietà del progetto.

momento anche le classi presenti nel file esterno al progetto saranno raggiungibili per il compilatore come quelle interne.

LA DIRECTORY BIN

Quando compiliamo un progetto Flex, il compilatore prepara nella directory Bin, sempre presente nella cartella principale del progetto, due files per noi importanti. Il primo è il file .swf che rappresenta la nostra applicazione, il secondo è uno stub .html per permette di lanciare l'applicazione previo controllo che sia presente il plug-in giusto del Flash Player, cioè la versione 9. Quando effettuiamo il deploy dell'applicazione, possiamo semplicemente copiare questa cartella avendo cura di eliminare i files che nel nome presentano "-debug" dato che questi rappresentano le versioni di debug dell'applicazione e di solito non è necessario portarli in "produzione". Oppure possiamo prendere spunto dal file .html per creare il nostro stub, ma in questo caso ricordiamo di copiare anche i files .js.

CONCLUSIONI

Bene. Di carne al fuoco ne abbiamo messa parecchia per una prima puntata. Vi lascio quindi all' installazione dell'ambiente e al rivedere i concetti che abbiamo affrontato fin qui. Seguendo i passi descritti sarete in grado di disegnare il primo pezzo della nostra applicazione di esempio. Troverete comunque i sorgenti allegati all'articolo. Nella prossima puntata poi, cominceremo ad aggiungere le funzionalità che andranno a costituire il nostro Music Shop.

Giuseppe Dattilo

L'AUTORE

Giuseppe Dattilo lavora nel campo dell'I.T. dal 1993. Ha partecipato a svariati progetti prevalentemente in ambito Windows, realizzando sia applicazioni Web-oriented che Desktop. Dal 2004 lavora utilizzando la piattaforma .NET e Visual C#. Chi volesse contattarlo può farlo all'indirizzo melnac@libero.it.

NETWORKING NATIVO IN JAVA

REALIZZARE UN'APPLICAZIONE CHE COMUNICHI ATTRAVERSO INTERNET O UN SERVER CHE SI METTA IN ASCOLTO SU UNA PORTA PUÒ ESSERE RELATIVAMENTE SEMPLICE. SCOPRIAMO QUALI SONO GLI STRUMENTI CHE IL LINGUAGGIO DI SUN CI METTE A DISPOSIZIONE

Questi ultimi dieci – quindici, nel campo della programmazione e non solo, hanno contribuito a formare quella che da tutti è conosciuta come l'era del networking. Quando poi si analizza la situazione dal lato degli sviluppatori si nota una cosa che lascia un pò spiazzati: esiste una "divisione" tra chi si occupa di applicazioni che gireranno su desktop e quelle che saranno ospitate su un web server (per intenderci le applicazioni web). Naturalmente questa non vuole essere una regola generale ma sono convito che se ci riflettete un po' su anche voi probabilmente vi sentirete più vicini ad una categoria piuttosto che all'altra. Quest'articolo ed i prossimi hanno proprio l'intento di trovare dei punti di contatto tra questi due mondi, facendo anche vedere come in alcuni casi risulti utile sfruttare l'approccio del networking per fare comunicare applicazioni stand-alone.
Parlando poi di networking non potremmo esimerci dal parlare di sicurezza ed in particolare focalizzeremo la nostra attenzione su SSH.

MODELLO ISO/OSI E TCP/IP

Il protocollo TCP/IP è lo standard "de facto" su cui si basa internet e grazie al quale la stessa internet ha potuto conoscere la sua grande espansione. Non è ovviamente questa la sede per parlare di questo protocollo. Libri e libri sono stati scritti, per non parlare della documentazione presente in rete, basti pensare alle RFC (*www.rfc.org*) che per gran parte definiscono proprio i vari protocolli.
La cosa che a noi interessa è il concetto di "strati", o all'inglese "layers", che sta alla base del funzionamento del TCP/IP. In effetti lo stesso nome TCP/IP deriva dagli ultimi due layers del protocollo stesso. Per rendere maggiormente chiaro questo concetto dobbiamo partire da modello ISO/OSI che concettualmente precede il modello TCP/IP.
La sigla OSI sta per **Open Systems Interconnection**, uno standard di comunicazione che stabilisce una pila di protocolli in 7 livelli, approvato nel 1978 dal International Organization for Standardization, il principale ente di standardizzazione internazionale, (ISO); per questo è meglio conosciuto come **Modello ISO/OSI**.
La figura 1 mostra la stratificazione prevista da ISO/OSI. Senza entrare nei dettagli potremmo sinteticamente dire che ogni layer individua un protocollo di comunicazione del livello medesimo, in modo tale si realizza una comunicazione per livelli.
In altre parole supponiamo che due nodi A e B vogliano comunicare, il livello n del nodo A può scambiare informazioni col livello n del nodo B ma non con gli altri: ciò permette di avvalersi di due fondamentali proprietà (forse una conseguenza dell'altra):

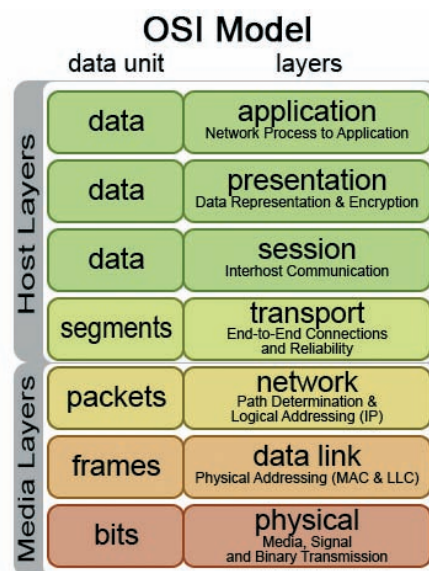



***Fig. 1:** rappresentazione dello stack di layers del modello ISO/OSI.*

1 - modularità del sistema.
2 - semplicità di implementazione e reimplementazione.
E' necessario naturalmente poi che ogni livello realizzi la comunicazione col livello immediatamente contiguo (sottostante o sovrastante). Sicché ISO/OSI

incapsula i messaggi di livello n in messaggi del livello n-1. Per fare un esempio mettiamo che A voglia inviare una mail a B: l'applicazione (liv. 7) di A propagherà il messaggio usando il layer sottostante (liv. 6), e che a sua volta userà il layer inferiore, fino ad arrivare alla comunicazione sul mezzo fisico. A sua volta dalla parte di B il messaggio verrà propagato dal liv. 1 fino al liv. 7. Il meccanismo fondamentale che ci interessa sottolineare e proprio la "divisione" dei compiti che una tale architettura permette. Ad esempio ci implementi un algoritmo per il protocollo di trasporto non deve preoccuparsi di come il pacchetto verrà poi instradato perché questo è compito del layer inferiore. In altre parole non dovrà mai chiedersi: ma questo pacchetto come arriverà a destinazione? Questo modo di ragionare sarà fondamentale un paio di paragrafi più avanti, quando cominceremo ad implementare la nostra prima applicazione che comunicherà attraverso TCP/IP.
Anche TCP/IP si basa sul medesimo concetto di stratificazione, proprio come il modello ISO/OSI, ma il suo stack di layers (**figura 2**) è differente. La prima differenza che salta all'occhio è sicuramente il numero di strati: 7 per il modello ISO/OSI, 5 per quello TCP/IP. Le differenze tra i due protocolli sono diverse, ma quella principale consiste nel fatto che nel TCP/IP il layer applicativo è esterno alla pila di protocolli (ovvero è una applicazione stand-alone che 'usa' TCP/IP per comunicare con altre applicazioni). Sarà proprio questo strato che andremo implicitamente ad implementare quando realizzeremo la nostra applicazione.

Fig. 2: Confronto fra lo stack di layers del modello ISO/OSI con quello TCP/IP.

"attacco" (infatti se digitate socket su google vi appariranno un sacco di prese per la corrente).
La prima volta questo meccanismo fu impiegato nella Berkeley Software Distribution (BSD) della University of California a Berkeley ed è stata sicuramente una della tappe miliari del networking.
Il nome socket ci rende bene l'idea di cosa esso sia e come funzionino. In effetti un socket è come una porta di comunicazione, concettualmente simile ad una presa elettrica: qualsiasi cosa progettata per comunicare tramite il protocollo standard TCP/IP, può collegarsi ad un socket e comunicare tramite questa porta di comunicazione, così come un qualsiasi apparecchio elettrico alimentato con la corrente 220 può collegarsi ad una presa elettrica e sfruttare la tensione che la rete di distribuzione elettrica mette a disposizione. Nel mondo dei socket potremmo dire che la rete di distribuzione è internet stessa ed invece dell'elettricità, nella rete viaggiano pacchetti TCP/IP. In questo modo i socket quindi ci forniscono un'ulteriore astrazione oltre a quella fornita da TCP/IP, che permette di far comunicare dispositivi diversi che utilizzano lo stesso protocollo. In altre parole i socket possono rappresentare il 6 livello del modello ISO/OSI che nel TCP/IP è assente (anche se è difficile effettuare un mappaggio così diretto 1:1).
Entriamo un po' più nel dettaglio del paradigma socket. Una delle tante informazioni che il livello TCP mantiene in ogni segmento sono i numeri delle applicazioni di porta d'origine e di destinazione ognuna delle quali codificata a 16 bits (vedi **figura 3**). In parole povere possiamo dire che TCP ha bisogno di sapere a quale porta sia destinato il pacchetto e da quale porta arrivi, in modo tale che possa ad esempio rispondere con un ACK a pacchetto ricevuto. In maniera analoga un pacchetto del livello IP porta con se l'indirizzo IP d'origine e quello di destinazione.

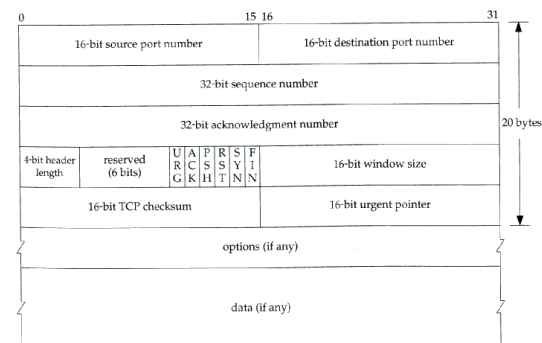



Fig. 3: Confronto fra lo stack di layers del modello ISO/OSI con quello TCP/IP.

SOCKET E TCP/IP

È arrivato quindi il momento di parlare di cosa siano i socket. Il termine socket in inglese significa "presa",

Un socket è costituito dal concatenamento di un indirizzo IP di host con il numero di porta di una specifica applicazione che risiede su quell'host. Quindi un socket identifica una coppia univoca

composta da host e applicazione, i cui numeri sono separati da due punti (:). Questo paradigma lo usiamo tutti i giorni senza accorgercene. Ad esempio una well-known port (una porta di default) è la 80 che si usa per il protocollo HTTP, se avete un web server installato puntate il browser all'indirizzo 127.0.0.1:80 e vi apparirà la main page del vostro server.

JAVA ED I SOCKET

A questo punto dovremmo avere acquisito qualche concetto chiaro: ci serve un host con un suo IP e in grado di comunicare attraverso una porta aperta sull'host. Ogni tipo di comunicazione prevede sempre almeno due attori. Nel nostro caso questo aspetto apparentemente banale pone una questione: è necessario che i due interlocutori comunichino sulla stessa porta contemporaneamente! Altrimenti chi manda il messaggio troverebbe la "porta chiusa" dell'altro e viceversa. Da qui nasce l'esigenza del paradigma client – server. Tenendo d'occhio la figura 4 vediamo qual è il ruolo di ciascuno dei due interlocutori.

La nostra prima applicazione sarà molto semplice:
1 – Un server aprirà una porta sul proprio IP address e resterà in ascolto su essa.
2 – Un client punterà a quell'indirizzo IP ed invierà dei messaggi sulla porta aperta dal server
3 – Il server non farà altro che rimandare il messaggio ricevuto al client. Chiameremo infatti questo server "EchoServer"
4 – Per convenzione il server chiuderà il socket, e quindi la porta, quando riceverà un messaggio vuoto. La **figura 4** sintetizza schematicamente quanto detto fin qui.

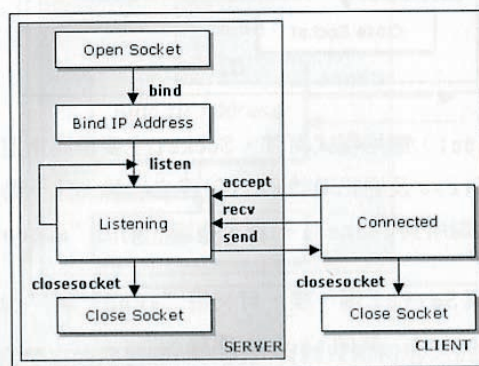



Fig. 4: schema del paradigma client-server nell'uso dei socket

IL SERVER

Naturalmente Java mette già a disposizione delle classi che implementano tali concetti, vediamo quindi prima di tutto il server.

La classe che Java mette a disposizione per assolvere alle funzioni previste per il server è la classe ServerSocket. Infatti tramite quest'ultima è possibile accettare connessioni dai client attraverso la rete su una ben determinata porta.

All'atto pratico quindi quando si vuole realizzare un server si possono seguire i seguenti tre punti:

1 - creare un oggetto di classe ServerSocket impostando al costruttore il numero di porta locale, cioè la porta in cui il server rimarrà in ascolto di richieste di connessioni.
2 - invocare il metodo accept della classe ServerSocket, in modo tale da attendere le connessioni da parte dei client.
3 - usare il socket ottenuto ad ogni connessione, per comunicare col client.

Veniamo quindi al codice del nostro EchoServer:

```
public class EchoServer implements Runnable {
        public static final int PORT = 4422;
        private boolean loop = true;
        private ServerSocket sSocket = null;
        private boolean running = false;
        ........
```

abbiamo specificato su un campo constante e pubblico sulla quale porta il server rimanga in ascolto, in modo tale che i client possano saperlo facilmente. Un altro membro è proprio sSocket che implementa le funzionalità del server vero e proprio.

Il perché si sia deciso che EchoServer implementi l'interfaccia Runnable sarà più chiaro tra poco, per ora limitiamoci ad analizzare il metodo run che questa prevede:

```
public void run() {
        try {
                sSocket = new
                        ServerSocket(PORT);
        } catch (IOException e) {
                e.printStackTrace();
                return;
        }
        Socket clientSocket = null;
        PrintWriter out = null;
        BufferedReader in = null;
        running = true;
        try {
                while (loop) {
                        clientSocket =
                        sSocket.accept();
                        out = new
        PrintWriter(clientSocket.getOutputStream(),true);
                        in = new
                        BufferedReader(new
```

```
    InputStreamReader(clientSocket.getInputStream()));
                    String line;
                    while ((line =
                in.readLine()) != null)
out.println("Server response: "+ line);

            }
        } catch (IOException e) {
            e.printStackTrace();
        }
        finally {
            if (out != null)
                out.close();
            try {
                if (in != null)
in.close();
                if(clientSocket
                    != null)
clientSocket.close();
                if(sSocket !=
                    null)
sSocket.close();
            } catch (IOException e)
                    {
e.printStackTrace();
            }
            running = false;
        }
        System.err.println("Server is
                    down");
    }
```

NOTE

un indirizzo IP è un sequenza di quattro byte che identifica univocamente un host all'interno di una rete. Ad esempio il localhost ha come indirizzo di default 127.0.0.1 (in notazione decimale naturalmente).

Cerchiamo di analizzare tale metodo passo per passo in modo tale da "ritrovarci" i punti enucleati precedentemente. In primo luogo è necessario istanziare un ServerSocket sulla porta che abbiamo scelto. Se questa operazione non andasse a buon fine sarebbe inutile proseguire ed infatti sul catch di IOException, dopo aver stampato l'errore, usciamo direttamente dal metodo.

```
        try {
            sSocket = new
                ServerSocket(PORT);
        } catch (IOException e) {
            e.printStackTrace();
            return;
        }
```

A questo punto dobbiamo prepararci a poter ricevere un connessione da parte di un client, ciò avviene qui:

```
        Socket clientSocket = null;
        PrintWriter out = null;
        BufferedReader in = null;

        running = true;
```

Infatti il metodo accept (invocato immediatamente dopo) della classe ServerSocket crea un oggetto Socket per ogni connessione. Il server potrà poi comunicare con il client attraverso lo stream di input e quello di output grazie al BufferedReader ed al PrintWriter. In particolare quindi abbiamo:

```
try {
        while (loop) {
            clientSocket =
            sSocket.accept();
            out = new
    PrintWriter(clientSocket.getOutputStream(),true);
            in = new
            BufferedReader(new
    InputStreamReader(clientSocket.getInputStream()));
            String line;
            while ((line =
        in.readLine()) != null)
out.println("Server response: "+ line);

        }
```

una volta accettata la connessione, "incapsuliamo" lo stream di input nel reader e quello di output nel writer. Fatto ciò, per quanto ci eravamo prefissi, non dobbiamo far altro che leggere quanto il client ci ha inviato e "rispedirglielo indietro" con il suffisso "*Server response:*". Come avrete sicuramente notato questa parte di codice è racchiusa all'interno di un ciclo while che ha come condizione di terminazione la variabile *loop*. Infatti come ogni buon server che si rispetti deve continuare a rimanere in ascolto finché non viene "messo giù". Faccio inoltre notare che il metodo accept rimane in attesa finché non riceve un messaggio quindi non dovete pensare che questo sia un ciclo continuo. Una volta uscito da questo loop è necessario chiudere i vari stream e socket, l'ordine corretto è il seguente:

```
        if (out != null)
            out.close();
        try {
            if (in != null)
in.close();
            if(clientSocket
                != null)
clientSocket.close();
            if(sSocket !=
                null)
sSocket.close();
            ...
```

E' importante ricordarsi che naturalmente bisogna evitare di chiudere i socket prima che vengano chiusi i loro relativi stream. Ora torniamo al comportamento da server che questa classe deve avere, ossia occupiamoci di quel loop che fa sì che il server

rimanga in attesa della prossima richiesta da parte del client. Come prima caratteristica è necessario che, così come si può avviare il servizio tramite il metodo run, allo stesso modo sia possibile fermarlo. Un'analisi grossolana ci potrebbe portare a dire: baste settare la variabile loop del ciclo while a false e siamo a posto. In realtà benché l'idea di fondo sia giusta, presa così non funzionerebbe, per due ragioni principali:

1 – la modifica del valore della variabile loop deve essere effettuata da un thread esterno a quello che ha lanciato run (appunto perché quest'ultimo è "bloccato" sul ciclo while).
2 – anche mettendo a false la variabile loop questo non ci garantirebbe l'uscita dal ciclo, infatti ricordiamo che la riga di codice *clientSocket = sSocket.accept()*; rimane in attesa della prossima richiesta. Risulta quindi necessario inoltre inviare anche un messaggio "fantoccio" che serva esclusivamente a "sbloccare" il ciclo da quel punto.

Il metodo stop del server risulta quindi essere:

```
public void stop() {
        loop = false;
        Socket dummySocket = null;
        PrintWriter writer = null;
        try {
                dummySocket = new
                Socket("127.0.0.1", PORT);
                writer = new
        PrintWriter(dummySocket.getOutputStream(),true);
                writer.println("");
        } catch (UnknownHostException
                                        e) {
                e.printStackTrace();
        } catch (IOException e) {
                e.printStackTrace();
        } finally {
                if (writer != null)
                        writer.close();
                if (dummySocket !=
                                        null)
                        try {
dummySocket.close();
                        } catch
(IOException e) {
                                e.printStackTrace();
                        }
        }
    }
```

Come si può notare la prima cosa che si fa è settare la variabile globale loop a false, dopo di che dobbiamo inviare quel famoso messaggio per sbloccare il server dal suo ciclo. Questo ci da anche lo spunto per parlare del lato client.

Quello che vogliamo fare non è altro che mandare un messaggio vuoto al server. Naturalmente è scontato che il server a cui dobbiamo inviare tale messaggio risieda sullo stesso host, per cui l'indirizzo a cui punterà il nostro dummySocket sarà 127.0.0.1. Una volta stabilita la connessione non resterà da fare altro che scrivere sul proprio stream di output in modo tale che "dall'altra parte" il server legga tale messaggio. In altre parole non stiamo facendo altro che vedere l'altra faccia della medaglia della risposta del server. Per scrivere su tale stream di output abbiamo al solito utilizzato un *PrintWriter*

PENSARE IN MULTI-THREAD

Abbiamo già accennato al fatto che un servizio di questo tipo non può "auto-stopparsi" da solo ma è necessario che un altro thread lo faccia invocando il metodo stop illustrato poc'anzi.
Se è vero quindi che dobbiamo utilizzare tale server necessariamente in un ambiente multi-thread sarà quindi anche indispensabile sapere quando il server è effettivamente attivo, cioè quando si è veramente collegato alla porta specificata e si è messo in ascolto. Questo è il significato della variabile running che viene impostata a true appena prima dell'ingresso al ciclo while ed a false appena usciti da quest'ultimo.
A questo punto dovrebbe cominciare ad essere chiaro perché si sia deciso di far implementare alla classe EchoServer l'interfaccia Runnable; così infatti sarà possibile lanciare tale classe come thread parallelo (vedremo poi come). Poter sapere quando il server è effettivamente attivo è di fondamentale importanza, infatti tutti i client dovranno attendere l'attivazione di quest'ultimo prima di iniziare qualsiasi comunicazione con esso. A tale scopo è stato implementato il metodo waitForRunnig():

```
public void waitForRunnig() {
        while(!isRunning())
                Thread.yield();
}
public boolean isRunning() {
        return running;
}
```

Questo metodo non fa altro che controllare ciclicamente lo stato della variabile running, terminando quando essa assume il valore true. È importante notare che non si tratta di un semplice ciclo while di "attesa" (cioè un ciclo con uno statement vuoto), bensì viene invocato il metodo statico *Thread.yield()*;. Esso, secondo le specifiche della stessa SUN MicroSystem, rappresenta un "consiglio" che dice alla virtual machine: <<io per adesso posso attendere, esegui pure gli altri thread >>.

Naturalmente non c'è alcuna garanzia che la virtual machina segua il nostro consiglio e si comporti così.

LA NOSTRA PRIMA COMUNICAZIONE VIA SOCKET

Dopo tanto parlare è finalmente giunto il momento di instaurare la nostra prima comunicazione client-server. Per semplicità faremo girare sia il client che il server sullo stesso host; ad ogni modo se avete la possibilità di disporre di due PC in rete potete tranquillamente far girare il server da una parte ed il client dall'altra. Cominciamo ad analizzare il codice:

```
public static void main(String[] args) throws
        IOException {
    EchoServer serverSocket = new
        EchoServer();
    Thread server = new
        Thread(serverSocket);
    server.start();
    serverSocket.waitForRunnig();
    .....
```

Prima di tutto è necessario istanziare un server ed incapsularlo all'interno di un oggetto thread, una volta lanciato quest'ultimo attenderemo che sia veramente attivo grazie al metodo waitForRunnig() precedentemente discusso.

```
    Socket echoSocket = null;
        PrintWriter out = null;
        BufferedReader in = null;

        try {
            echoSocket = new
                Socket("127.0.0.1", PORT);
            out = new
    PrintWriter(echoSocket.getOutputStream(), true);
            in = new BufferedReader(new
                InputStreamReader(
echoSocket.getInputStream()));
        } catch (UnknownHostException e) {
            System.err.println("Don't know about
                host: taranis.");
            System.exit(1);
        } catch (IOException e) {
            System.err.println("Couldn't get I/O for "
                + "the connection to:
                localhost.");
            System.exit(1);
        }
```

Una volta che il server risulta attivo possiamo collegare il nostro socket-client e comunicare attraverso i suoi stream come visto in precedenza.

```
        BufferedReader stdIn = new
            BufferedReader(
                new
            InputStreamReader(System.in));
        String userInput;
        while ((userInput =
            stdIn.readLine()) != null) {
            if(userInput.equals(""))
            {
serverSocket.stop();
                break;
            }
            out.println(userInput);
System.out.println(in.readLine());
            }

        out.close();
        in.close();
        stdIn.close();
        echoSocket.close();

    }
```

In particolare la nostra applicazione userà lo standard input (quello della console per intenderci) per comunicare con il server. Inoltre abbiamo stabilito la convenzione che un semplice invio senza alcun messaggio stoppi il server. Di seguito è riporta tale comportamento:

```
ciao
Server response: ciao
sono il client
Server response: sono il client
Server is down
```

CONCLUSIONI

In questo primo articolo abbiamo avuto modo di cominciare a capire un po' meglio il funzionamento del protocollo TCP/IP, di come debba funzionare un'architettura client-server ed infine quali strumenti Java mette a disposizione per il networking. Diciamoci la verità, un server che ci risponde quello che gli abbiamo inviato non è proprio il massimo dell'utilità. Nei prossimi articoli partiremo dalle conoscenze fin qui acquisite per realizzare un'applicazione di instant messagging, vedremo cosa passa realmente attraverso la rete con uno sniffer ed affronteremo quindi anche il problema della necessità di comunicazioni criptate. Spero di aver solleticato la vostra curiosità per i prossimi articoli.

Andrea Galeazzi

L'AUTORE

Laureato in ingegneria elettronica presso l'università Politecnica delle Marche, lavora presso il reparto R&D della Korg S.p.A. Nei limiti della disponibilità di tempo risponde all'indirizzo andreagaleazzi@fsfe.org

UN PAINTBRUSH PORTABILE CON C++

NONOSTANTE WXWIDGETS DISPONGA DI UN NUTRITO PARCO DI CONTROLLI PRECOSTRUITI, A VOLTE È NECESSARIO SPORCARSI LE MANI E DISEGNARE GRAFICAMENTE SULLE FINESTRE. PER IMPARARE UTILIZZEREMO UN ESEMPIO PRATICO PIUTTOSTO DIVERTENTE

Benvenuti al nostro quarto appuntamento con wxWidgets. Nei numeri precedenti abbiamo imparato a gestire le finestre, a rispondere agli eventi, e a disporre i controlli secondo layout avanzati.
In questo numero faremo numerosi passi avanti, imparando alcuni elementi fondamentali (non sappiamo ancora come creare un menù o una barra degli strumenti!), e dedicando grande attenzione al disegno diretto sulle finestre e alla gestione dell'input via mouse.
Come banco di prova per le nostre scoperte, creeremo la rudimentale applicazione ***wxSketch***, che potete ammirare in *Figura 1*. Pronti? Cominciamo!

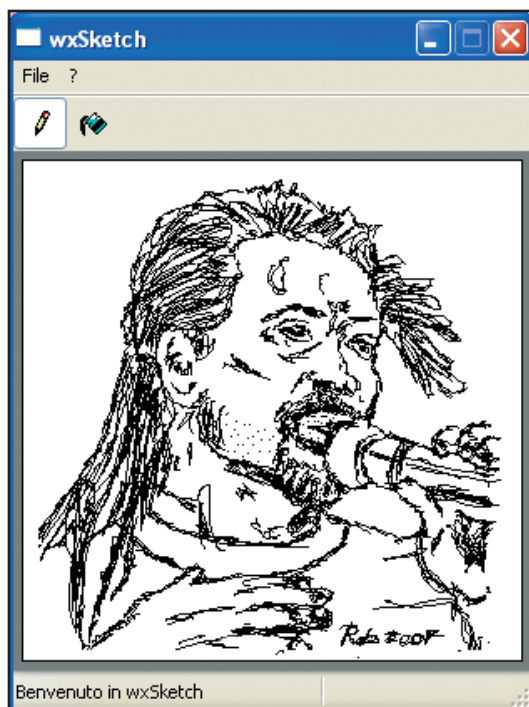



Fig. 1: Schema del nostro database

L'APPLICAZIONE

Come al solito, definiamo una classe derivata da *wxApp*, che si occupi di creare il frame principale. Stavolta, però, quest'ultimo avrà una limitazione: non sarà ridimensionabile dinamicamente. Questo ci renderà le cose più semplici, e ci insegnerà come creare frame con stili personalizzati.

```
class MainFrame : public wxFrame
{
public:
    MainFrame();
private:
    ConstructMenu();
}

class wxSketchApp : public wxApp
{
public:
    bool OnInit() {
        MainFrame* frame = new MainFrame();
        frame->Show();
        return true;
    }
};

MainFrame::MainFrame() :
    wxFrame(0, wxID_ANY, _T("wxSketch"),
    wxDefaultPosition, wxDefaultSize,
    wxMINIMIZE_BOX | wxSYSTEM_MENU |
                    wxCAPTION | wxCLOSE_BOX) {
    ConstructMenu()
}
```

La prima cosa che salta all'occhio è il lungo costruttore di *wxFrame*, richiamato con ben cinque argomenti: genitore, ID, titolo, posizione (*wxDefaultPosition* indica che non siamo interessati a specificarne una), dimensione (idem), e stile. Quest'ultimo può essere composto da più flag, in mancanza dei quali viene assunta la combinazione predefinita *wxDEFAULT_FRAME_STYLE*, ovverosia:

```
wxMINIMIZE_BOX | wxMAXIMIZE_BOX |
```

```
wxRESIZE_BORDER | wxSYSTEM_MENU |
wxCAPTION | wxCLOSE_BOX | wxCLIP_CHILDREN
```

Per il nostro frame abbiamo eliminato da questi flag quelli relativi al resize, ottenendo una finestra impossibile da ridimensionare o ingrandire a tutto schermo (mentre è sempre possibile ridurla a icona).

MENU

La seconda cosa che salta all'occhio nel codice precedente è la funzione privata *ConstructMenu*, dichiarata in *MainFrame* e richiamata nel suo costruttore. Al suo interno costruiremo la barra dei menu, limitandoci a 'File -> Esci' e '? -> Informazioni su wxSketch'.

```
void MainFrame::ConstructMenu() {
    wxMenuBar* menuBar = new wxMenuBar();

    wxMenu* fileMenu = new wxMenu();
    wxMenu* helpMenu = new wxMenu();

    menuBar->Append(fileMenu, _T("&File"));
    menuBar->Append(helpMenu, _T("&?"));

    //File --> Esci
    wxMenuItem* exitItem = new wxMenuItem(
            fileMenu, wxID_EXIT, _T("&Esci\tCTRL+X"),
      _T("Termina l'applicazione")
    );

    fileMenu->Append(exitItem);

    //? --> Informazioni su wxSketch
    wxMenuItem* aboutItem = new wxMenuItem (
      helpMenu, wxID_ABOUT,
     _T("&Informazioni su wxSketch\tF1"),
     _T("Chi ha creato questo programma, e perché?")
    );

    helpMenu->Append(aboutItem);

    SetMenuBar(menuBar);
}
```

Analizziamo il codice. Per prima cosa, bisogna creare sullo heap una nuova barra dei menu (*menuBar*, istanza di ***wxMenuBar***), e associarla alla finestra richiamando la funzione membro ***wxFrame::SetMenuBar***. In questo modo il frame gestirà la nuova barra automaticamente, e quello che è più importante, esternamente all'area di disegno.
Ora possiamo cominciare a riempire la barra con una serie di menu (come *fileMenu* e *helpMenu* istanze di ***wxMenu***), che possono essere aggiunti tramite la funzione membro *Append*.
I menu sono a loro volta dei contenitori, ai quali possono essere aggiunti (sempre tramite una funzione *Append*) delle voci (come *exitItem* o *aboutItem*, istanze di ***wxMenuItem***).
Il costruttore di un *wxMenuItem* può richiedere quattro argomenti: il menu padre, l'identificatore, il testo, e una descrizione più lunga. Nel nostro caso abbiamo usato gli identificatori predefiniti ***wxID_ABOUT***, e ***wxID_EXIT***, il che dovrebbe essere considerato un obbligo, più che una comodità. Su alcune piattaforme, infatti, le voci di menu predefinite interagiscono in maniera speciale con la finestra o col sistema – nel nostro caso, ad esempio, sotto Mac OS le due voci verranno rimosse dal menu dell'applicazione e inserite in quello di sistema.
Notate anche le tabulazioni alla fine del testo delle voci: servono a definire una scorciatoia via tastiera. In questo modo, l'utente potrà uscire premendo i tasti CTRL+X, e richiedere informazioni premendo il tasto F1.

GESTORI DEI MENU

Ora la nostra applicazione mostra un menu, ma cliccando sulle voci non succede niente! Per associare delle azioni ad un menu, si usano gli strumenti che abbiamo già visto per la gestione dei messaggi: le Event Table. Facciamo qualche piccola aggiunta a *MainFrame*:

```
class MainFrame : public wxFrame
{
  //[...] resto dell'implementazione [...]

    protected:
    void OnAbout(wxCommandEvent& event);
    void OnQuit(wxCommandEvent& event) {Close()};

    DECLARE_EVENT_TABLE()
};

BEGIN_EVENT_TABLE(MainFrame, wxFrame)
    EVT_MENU(wxID_ABOUT, MainFrame::OnAbout)
    EVT_MENU(wxID_EXIT, MainFrame::OnQuit)
END_EVENT_TABLE()   EVENT_TABLE
```

Abbiamo implementato le due funzioni *OnAbout* e *OnQuit* e una Event Table per associarvi i messaggi del menu, per mezzo della macro *EVT_MENU*. Notate che queste funzioni richiedono il passaggio di un *wxCommandEvent*, esattamente come quelle dei pulsanti (e in effetti, un menù non è che un complesso "indice di pulsanti").

NOTA

BIBLIOSITOGRAFIA

Per tutti i dettagli su pennelli, penne e classi immagine (e per tutto il resto), fate riferimento alla documentazione ufficiale (http://wxwidgets.org/docs/).

Se, come me, preferite i libri di carta, Cross-Platform Gui Programming with wxWidgets, scritto da Julian Smart (lo stesso autore della libreria), è un buon punto di partenza e un primo punto di riferimento per il framework.

Infine, ricordate che wxWidgets è open source, e spesso il codice sorgente è la migliore delle documentazioni!

La definizione delle due funzioni è banale: OnQuit si limita a chiudere il frame (terminando così l'applicazione, come abbiamo visto nel secondo appuntamento). *OnAbout* richiama un ***wxMessageBox*** con il solito disclaimer.

MENU, BITMAP E WXARTPROVIDER

Il nostro menu è un po' troppo spoglio. Quando ha un senso farlo, infatti, bisognerebbe sempre associare alle voci un'icona descrittiva – in modo da aiutare il colpo d'occhio dell'utente e non obbligarlo a pensare a cosa sta premendo.
WxWidgets permette di farlo tramite la funzione membro ***wxMenuItem::SetBitmap***, che prende come parametro un oggetto di tipo ***wxBitmap***. Ma cos'è un oggetto *wxBitmap*, e come si crea? Bella domanda.
wxBitmap è una delle classi che wxWidgets usa per la gestione delle immagini (ce ne sono anche altre, come ad esempio ***wxIcon, wxCursor*** e ***wxImage***). Si tratta di una rappresentazione interna di una bitmap, assolutamente dipendente dal singolo Sistema Operativo. Ogni SO, infatti, gestisce le immagini in memoria a suo modo, e wxBitmap rappresenta un livello di indirettezza in grado di superare l'ostacolo fornendo un'interfaccia comune e coerente.
Nel nostro caso, vogliamo ottenere un piccolo *wxBitmap* (di 16x16 pixel), da associare alla voce "Esci". Possiamo farlo facilmente grazie alla funzione membro statica ***wxArtProvider::GetBitmap***.

```
const wxPoint IconSize(16, 16);

exitItem->SetBitmap(
  wxArtProvider::GetBitmap(wxART_QUIT,
                                      wxART_MENU, IconSize)
);
```

Il metodo richiede tre parametri: l'immagine che vogliamo, il tipo di client in cui vogliamo inserirla (in questo caso vogliamo inserirla in un menu), e la grandezza della bitmap. Se *wxArtProvider* "conosce" l'immagine che stiamo chiedendo, ce la restituirà sotto forma di *wxBitmap*.

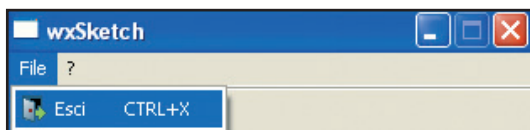



Fig. 2: Le due item con le icone wxART_QUIT e wxART_ABOUT

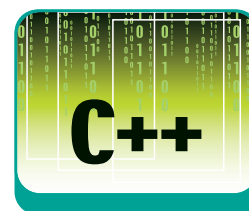
wxArtProvider può fornire bitmap predefinite per tutte le icone più comuni: nel nostro caso, ad esempio, possiamo usare gli identificatori *wxART_QUIT* (per *exitItem*) e *wxART_ABOUT* (per *aboutItem*) per ottenere le belle icone mostrate in Figura 2. Grazie a *wxArtProvider*, non solo possiamo ottenere delle icone "gratis", ma siamo soprattutto sicuri che queste saranno in accordo con il *look'n feel* del Sistema Operativo per il quale compileremo il programma.
Derivando da *wxArtProvider*, infine, è possibile ridefinire completamente le icone, in modo da creare un intero insieme di temi che possono essere cambiati dinamicamente.

LA BARRA DI STATO

Un altro elemento irrinunciabile nelle applicazioni è la barra di stato. Non viene notata quanto i menu, ma gli utenti vi fanno comunque grande riferimento, spesso senza neanche accorgersene. Una barra di stato può contenere informazioni circa lo stato corrente del programma (ad esempio: "quale pixel dell'immagine sta puntando il cursore?" oppure "Il documento è stato salvato correttamente?"), oppure fornire descrizioni sugli elementi puntati dall'utente.
Le informazioni sono tante che spesso è necessario suddividere la barra in più sezioni, e a volte anche a inserire veri e propri controlli al suo interno.
Nel nostro esempio, ci limiteremo a creare una barra di stato con due sezioni: una per i suggerimenti, e l'altra per indicare la posizione del mouse sulla tela. Ecco la funzione *ConstructStatusBar*, che richiameremo dal costruttore di *MainFrame*.

```
void MainFrame::ConstructStatusBar() {
    wxStatusBar* statusBar = new wxStatusBar(this);
    SetStatusBar(statusBar);

    int widths[] = {-3, -2};
    statusBar->SetFieldsCount(2, widths);
    statusBar->SetStatusText(_T("Benvenuto in
                                          wxSketch"), 0);
}
```

Come il menu, un frame sa gestire automaticamente una (e una sola) barra di stato: un oggetto di tipo wxStatusBar che va costruito sullo heap e affidato alla finestra con la funzione membro ***wxFrame::SetStatusBar***.
Dopodiché è necessario indicare le sezioni in cui la si vuole dividere, e le dimensioni di ciascun campo (il vettore *widths*). Questo può essere indicato in pixel (ad esempio {100, 200}), ma –

WXIMAGE

wxImage è una classe fondamentale per la gestione della grafica in wxWidgets, dal momento che permette di gestire un immagine con molti tipi di effetti e operazioni, e soprattutto la rappresenta in modo indipendente dal Sistema Operativo.

come abbiamo visto nella scorsa puntata – il dimensionamento assoluto è fonte di guai. Nel nostro caso, non sappiamo quale sia il font di sistema, quanto spazio potrebbero prendere i messaggi, o quanto sarà effettivamente larga la nostra finestra.
Per questo è meglio usare un dimensionamento proporzionale, che viene espresso con numeri negativi. Nel nostro caso, il primo campo occuperà i 3/5 dello spazio disponibile, e il secondo i rimanenti 2/5.
Una volta dichiarato l'array, possiamo richiamare la funzione membro ***wxStatusBar::SetFieldsCount***, indicandole che vogliamo due campi (primo parametro), con dimensioni indicate in widths (secondo parametro).
Con ***wxStatusBar::SetStatusText*** possiamo passare un testo (primo parametro) da associare ad uno dei campi (il cui indice è passato nel secondo parametro).
Il primo dei campi, infine, ha anche una funzione particolare: ricordate quando per creare le voci dei menu abbiamo passato un quarto parametro al costruttore di *wxMenuItem*? Si tratta di una stringa di descrizione, che viene usata internamente da wxWidgets per il primo campo della barra di stato.
Ogni volta che l'utente muoverà il puntatore su una delle voci del menu, la barra mostrerà automaticamente la relativa descrizione.

NOTA

ALTERNATIVE

Invece dell'approccio JDBC che è stato utilizzato nell'articolo per realizzare le classi DAO, è possibile utilizzare dei framework opensource di persistenza come Hibernate
http://www.hibernate.org
iBatis
http://ibatis.apache.org/
db4o
http://www.db4o.com/
pBeans
http://pbeans.sourceforge.net/
JPOX
http://www.jpox.org/

LA TELA

Sistemati gli elementi di contorno, è ora di creare la "tela" che servirà per il disegno. Sarà una classe che chiameremo Canvas, e che deriveremo direttamente da *wxWindow*:

```
class Canvas : public wxWindow {
public:
    Canvas(MainFrame* parent);
protected:
    MainFrame* parent_;
    static const wxSize Size;
};
const wxSize Canvas::Size(300, 300);
```

Una tela avrà una dimensione fissa, stabilita da *Canvas::Size*, e sarà creata direttamente nel costruttore di *MainFrame*:

```
void MainFrame::ConstructCanvas() {
    wxBoxSizer* sizer = new
                        wxBoxSizer(wxVERTICAL);
    SetSizer(sizer);
    canvas_ = new Canvas(this);
    sizer->Add(canvas_, 0, wxALL, 5);
    Fit();
    canvas_-
        >SetCursor(wxCursor(wxCURSOR_PENCIL));
}
```

La creazione ricalca fedelmente quanto abbiamo visto nella scorsa puntata. L'unica aggiunta è la chiamata alla funzione membro ***Fit***, che serve a modificare le dimensioni del frame in modo che si adattino perfettamente al contenuto del sizer. In questo modo eviteremo che il frame mostri dello spazio inutilizzato. La nostra applicazione apparirà (sotto windows) come mostrato in figura 3.

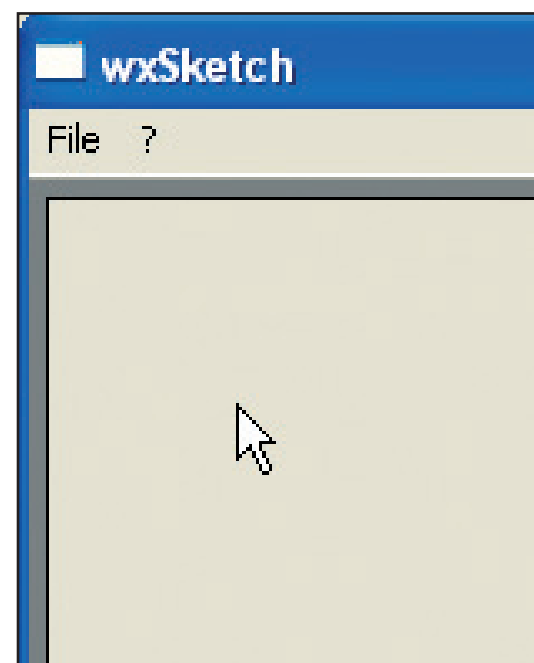



Fig. 3: La barra di stato mostra la posizione del cursore sulla tela.

Nell'ultima istruzione impostiamo la forma del puntatore del mouse su quello di una matita creando un oggetto **wxCursor** predefinito. Come potete vedere dalla figura 3, non sempre queste icone sono presenti nel Sistema Operativo.

CATTURIAMO IL TOPO

A dir la verità, in figura 3 è presente qualcosa in più: il secondo campo della barra di stato mostra le coordinate del cursore sulla tela. Per ottenere quest'effetto, possiamo definire queste funzioni membro in MainFrame:

```
void MainFrame::DisplayMousePosition(const
                                     wxPoint& p) {
    wxString str;
    str.Printf(_T("x=%d   y=%d"), p.x, p.y);
    GetStatusBar()->SetStatusText(str, 1);
```

```
}

void MainFrame::HideMousePosition() {
    GetStatusBar()->SetStatusText(_T(""), 1);
}
```

Ora possiamo richiamare *DisplayMousePosition* quando il cursore si trova in una data posizione, e *HideMousePosition* quando non vogliamo rappresentarla (ad esempio, quando il mouse è fuori dalla tela).
Per usare queste funzioni, però, dobbiamo imparare a catturare gli eventi del mouse! Espandiamo un po' la classe Canvas:

```
class Canvas
    //[...] resto dell'implementazione [...]
protected:
    void OnMouseMove(wxMouseEvent& event);
    void OnMouseLeave(wxMouseEvent& event);
    void OnMouseUp(wxMouseEvent& event);
    void OnMouseDown(wxMouseEvent& event);
    DECLARE_EVENT_TABLE()
};
BEGIN_EVENT_TABLE(Canvas, wxWindow)
    EVT_MOTION(Canvas::OnMouseMove)
    EVT_LEAVE_WINDOW(Canvas::OnMouseLeave)
    EVT_LEFT_DOWN(Canvas::OnMouseDown)
    EVT_RIGHT_DOWN(Canvas::OnMouseDown)
    EVT_LEFT_UP(Canvas::OnMouseUp)
    EVT_RIGHT_UP(Canvas::OnMouseUp)
END_EVENT_TABLE()
```

Come al solito, si tratta semplicemente di definire un'event table con voci appropriate, e delle funzioni. Nel nostro caso cattureremo il movimento del mouse sulla tela (*onMouseMove, EVT_MOTION*), il momento in cui il mouse uscirà dalla tela (*OnMouseLeave, EVT_LEAVE_WINDOW*), e la pressione/rilascio di uno dei due pulsanti del mouse (come vedete, è del tutto lecito associare più eventi allo stesso gestore).
L'argomento passato ai gestori è di tipo *wxMouseEvent*, e contiene utili informazioni sullo stato del cursore:

```
void Canvas::OnMouseMove(wxMouseEvent& event)
                                            {
    parent_-
            >DisplayMousePosition(event.GetPosition());
}

void Canvas::OnMouseLeave(wxMouseEvent& event)
                                            {
    parent_->HideMousePosition();
}
```

Grazie alla funzione membro *wxMouseEvent::GetPosition()* abbiamo potuto passare facilmente la posizione del cursore al frame.

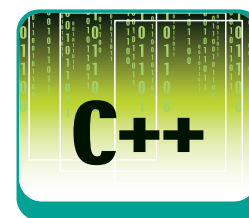
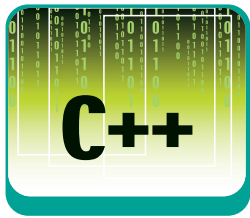
DIPINGIAMO SULLA TELA

Ora ci rimangono gli altri gestori per dipingere sulla tela. Per cominciare simuleremo il comportamento dello strumento "matita": ogni volta che il mouse si muove registreremo le sue coordinate in *Canvas::lastPosition_*, e congiungeremo con una linea questo punto a quello precedente.
Per riuscire nell'intento dobbiamo imparare a tracciare una linea su una finestra. Ogni Sistema Operativo usa le sue API e i suoi metodi, ma wxWidgets ci fornisce una via coerente: i **contesti di dispositivo** (o *Device Context*, o *DC*).
Un DC offre un insieme di funzioni per il disegno e la manipolazione della grafica su un contesto: wxWidgets offre molti DC differenti: ***wxClientDC*** per disegnare su una finestra, ***wxPrinterDC*** per disegnare su un documento di stampa, eccetera… Nonostante la loro eterogeneità, tutti offrono lo stesso insieme di operazioni, dal momento che derivano dalla classe ***wxDC***.

```
void Canvas::OnMouseDown(wxMouseEvent& event){
    lastPosition_ = event.GetPosition();
}
void Canvas::OnMouseMove(wxMouseEvent& event){
    // [...] resto dell'implementazione [...]
    if (event.Dragging()) {
        wxClientDC dc(this);
        dc.DrawLine(lastPosition_, event.GetPosition());
        lastPosition_ = event.GetPosition();
    }
}
```

In *MouseMove* controlliamo prima se l'utente sta premendo il pulsante (grazie a *Dragging*, una delle tante funzioni utili di *wxMouseEvent*), e in questo caso disegniamo la linea e procediamo all'aggiornamento. Come vedete, usiamo un ***wxClientDC***, che va costruito sullo stack passandogli un puntatore alla finestra su cui si vuole disegnare. A questo punto possiamo usare una miriade di funzioni membro del DC per disegnare sulla tela: richiamando ***DrawLine***, per esempio, viene immediatamente tracciata sulla finestra una linea.

IL PROBLEMA DEL RIDISEGNO

Ora il nostro *wxSketch* comincia ad essere interessante da usare. Se provate a giocarci per un

po', vi accorgerete di un problema molto grave: ciò che disegnate è molto precario. Infatti, basta ridurre l'applicazione a icona e ripristinarla, per perdere tutto!
Per capire il perché di questo comportamento, bisogna comprendere come avviene il disegno di una finestra in wxWidgets.
Ogni volta che il framework lo ritiene necessario (ad esempio quando una finestra è stata spostata e ha scoperto una parte dell'area di disegno), viene lanciato un evento *Erase*, e successivamente un evento Paint. Questi hanno lo scopo rispettivamente di cancellare tutto ridisegnando lo sfondo (il comportamento predefinito è sostituire l'area da cancellare con un bel rettangolo con il colore di ***GetBackgroundColor***), e di disegnare il contenuto vero e proprio della finestra (il comportamento predefinito è nullo).
Quando disegniamo le linee in un *ClientDC*, queste vengono impresse sull'area di disegno ma non vengono memorizzate da nessuna parte, e pertanto vengono perse ad ogni ridisegno.
Per riuscire a risolvere il problema, quindi, dovremo ridefinire questi eventi:

```
BEGIN_EVENT_TABLE(Canvas, wxWindow)
    //[...] Resto della Event Table [...]
    EVT_PAINT(Canvas::OnPaint)
EVT_ERASE_BACKGROUND(Canvas::OnEraseBackground)
END_EVENT_TABLE()
```

Purtroppo, però, il disegno delle linee avviene *esternamente* a questi eventi, pertanto prima di definirne i gestori dovremo stabilire un piano d'azione.

WXBITMAP E WXMEMORYDC

L'elemento principale del piano è questo: quando l'utente solleva il pulsante del mouse, copiamo tutto ciò che abbiamo disegnato sulla tela in memoria. A questo scopo dotiamo il Canvas di una *wxBitmap* grande quanto la tela.

```
class Canvas //[...] resto dell'implementazione [...]
protected:
    wxBitmap bitmap_;
};
Canvas::Canvas(MainFrame* parent) :
    parent_(parent), wxWindow(parent, wxID_ANY,
                              wxDefaultPosition, Size),
    bitmap_(Size.GetX(), Size.GetY()) {

    wxMemoryDC dc(bitmap_);
    dc.SetBrush(*wxWHITE_BRUSH);
    dc.DrawRectangle(0, 0, Size.GetX(), Size.GetY());
}
```

L'analisi di *Canvas::Canvas* è molto interessante. Come vedete, abbiamo creato la bitmap semplicemente richiamandone il costruttore e passandogli le dimensioni.
Quindi, all'interno del corpo del costruttore, *bitmap_* è già perfettamente creata e pronta per il disegno. Ma come si disegna su una *wxBitmap*? Con un DC apposito: **wxMemoryDC**.
Costruito il DC (passandogli come argomento la nostra bitmap), possiamo disegnare un bel rettangolo bianco che copra tutta l'area.
A questo punto passiamo alla seconda fase del piano. Quando l'utente solleva il pulsante del mouse, copiamo il disegno sulla tela all'interno della bitmap.
Per farlo otteniamo prima un *wxClientDC* della tela, quindi un *wxMemoryDC* della bitmap, e poi usiamo la funzione ***wxDC::Blit***, che si occupa di copiare pezzi di immagine da un DC all'altro.

```
void Canvas::OnMouseUp(wxMouseEvent& event) {
    CommitBitmap();
}
void Canvas::CommitBitmap() {
    wxMemoryDC bitmapDC(bitmap_);
    wxClientDC dc(this);
    bitmapDC.Blit(0,0, Size.GetX(), Size.GetY(), &dc,
                  0, 0);
}
```

EVENTI PAINT E ERASE

L'ultima fase del piano a questo punto è ovvia: ridefiniamo gli eventi *EraseBackground* e *Paint*, in modo che in un eventuale refresh venga riversato sulla tela il contenuto di *wxBitmap*. Niente più fastidiose cancellature!

```
void Canvas::OnEraseBackground(wxEraseEvent&
                               event) {}
void Canvas::OnPaint(wxPaintEvent& event) {
    wxMemoryDC bitmapDC(bitmap_);
    wxPaintDC dc(this);
    dc.Blit(0,0, Size.GetX(), Size.GetY(), &bitmapDC,
            0, 0);
}
```

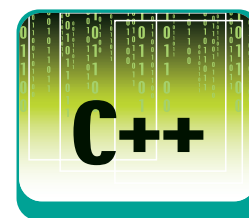
Come vedete, abbiamo definito *OnEraseBackground* con un corpo nullo. Questo impedirà che sfarfalli sullo schermo quell'orribile sfondo grigio in Figura 3. Il codice di *OnPaint* sembra quello di *CommitBitmap* visto allo specchio (infatti stavolta si va *dalla* bitmap *alla* tela). In effetti, l'unica differenza è l'uso di una nuova classe DC:

wxPaintDC. Un *wxPaintDC* è un DC speciale che va usato ***soltanto*** nel gestore di *EVT_PAINT*. Questo perché questa fase del disegno è molto delicata in alcuni Sistemi Operativi (come Windows), e un normale *wxClientDC* potrebbe generare problemi di ricorsione infinita. *wxPaintDC* si occupa di bloccare il ridisegno della finestra, e di sbloccarlo alla sua distruzione (chi ha seguito i nostri appuntamenti sul RAII sentirà suonare un campanello). Questa è la ragione per cui un oggetto *wxPaintDC* **va sempre e comunque istanziato** all'interno di un gestore di *EVT_PAINT*, anche se non si ha alcuna intenzione di usarlo.

LA BARRA DEGLI STRUMENTI

A questo punto ne sappiamo abbastanza per creare una serie di strumenti, oltre alla semplice matita. Questo passo ci permetterà di studiare come creare una barra degli strumenti in wxWidgets.

```
const int MainFrame::ToolID_Pencil = 0;
const int MainFrame::ToolID_FloodFill = 1;
void MainFrame::MainFrame()
    // [...] resto dell'implementazione [...]
    wxImage::AddHandler(new wxPNGHandler);
    ConstructToolBar();
}
void MainFrame::ConstructToolBar() {
    wxBitmap bmp;
    wxToolBar* toolBar = CreateToolBar();
    toolBar->SetToolBitmapSize(wxSize(25,25));
    bmp.LoadFile(_T("Matita.png"),
                                        wxBITMAP_TYPE_PNG);
    toolBar->AddRadioTool(
        ToolID_Pencil, _T("Matita"), bmp,wxNullBitmap,
        _T("Matita"), _T("Traccia un segno sulla tela"));
    bmp.LoadFile(_T("Riempimento.png"),
                                        wxBITMAP_TYPE_PNG);
    toolBar->AddRadioTool(
        ToolID_FloodFill, _T("Riempimento"), bmp,
                                              wxNullBitmap,
        _T("Riempi"), _T("Riempie uno spazio chiuso"));
    toolBar->Realize();
    SetToolBar(toolBar);
}
```

Analisi del codice: Un frame può gestire uno (e un solo) oggetto di tipo ***wxToolbar*** (è possibile aggiungerne altri, ma andranno gestiti a mano). La barra predefinita va creata con la funzione membro ***wxFrame::CreateToolBar()***.
Questa volta non abbiamo a disposizione delle icone predefinite per gli strumenti di disegno, quindi non possiamo affidarci a *wxArtProvider*. Questo ci dà l'occasione di vedere come caricare dei file in una *wxBitmap*. In questo caso abbiamo dei file PNG, pertanto la prima cosa da fare è attivare **l'handler** che insegnerà a wxWidgets a trattare con questo tipo di file. Una ToolBar può essere riempita di pulsanti di vario tipo, e perfino di controlli! In questo caso, scegliamo dei ***RadioTool***, ovvero pulsanti che permettono una selezione mutuamente esclusiva. Oltre a un ID univoco e al nome dello strumento, la funzione richiede la bitmap da disegnare (caricata in precedenza con *LoadFile*), un "aiuto breve" (che servirà per i tooltip), e un "aiuto lungo" analogo a quello dei menù, che finirà sulla barra di stato. Alla fine, è **fondamentale** richiamare la funzione ***wxToolbar::Realize***, che si occupa di creare la barra internamente. Se non lo fate, infatti, alcuni Sistemi Operativi (ad esempio Windows) non visualizzeranno alcun pulsante! Una volta associata al frame, tramite ***wxFrame::SetToolBar***, avremo a disposizione una bellissima barra degli strumenti, proprio come appare in figura 1.

WXCOLOR, PENNE E PENNELLI

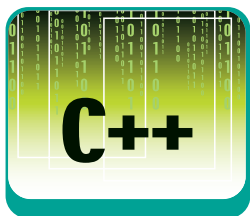
A questo punto, dobbiamo ritoccare i gestori del mouse, differenziando i messaggi per la matita da quelli per il riempimento. Per farlo possiamo sfruttare la funzione membro ***wxToolBar::GetToolState***, che indica se un dato RadioTool è correntemente selezionato.

```
void Canvas::OnMouseMove(wxMouseEvent& event)
                                              {
    //[...] resto dell'implementazione [...]
    if (parent_->GetToolBar()-
                    >GetToolState(ToolID_Pencil)) {

        //[...] implementazione della matita [...]
    }
}
```

Lo strumento *riempimento* è molto semplice da realizzare, dal momento che agisce soltanto al rilascio del tasto del mouse.

```
void Canvas::OnMouseUp(wxMouseEvent& event) {
    if (parent_->GetToolBar()->GetToolState(1)) {
        wxClientDC dc(this);
        dc.SetBrush(wxBrush(wxColour(_T("Black")),
                                  wxBDIAGONAL_HATCH));
        wxColour col;
        dc.GetPixel(event.GetPosition(), &col);

        dc.FloodFill(event.GetPosition(), col);
    }
    //[...] resto dell'implementazione [...]
}
```

Il codice è piuttosto semplice: per prima cosa prendiamo il colore del pixel su cui è posizionato il mouse (con ***wxDC::GetPixel***). Poi, richiamiamo la funzione ***wxDC::FloodFill***, che riempie un area contigua fintantoché trova il colore passato dal secondo argomento.

wxColour è la classe che wxWidgets usa per definire i colori. Un colore può essere ottenuto con GetPixel, oppure può essere creato con diversi costruttori – è possibile passare una terna Rosso/Verde/Blu (*wxColour(0,0,255)*, per il blu), oppure una *wxString* contenente il nome del colore (*wxColour(_T("Blue"))*), oppure ancora con delle macro che richiedano l'oggetto alla classe (come *wxBLUE*, che sta per *wxStockGDI::GetColour(wxStockGDI::COLOUR_BLUE)*).

Per impostare una modalità di riempimento, invece, abbiamo usato un oggetto ***wxBrush***, che può essere costruito in molti modi. Qualche paragrafo fa, ad esempio, abbiamo usato ***wxWHITE_BRUSH***, una macro che sta per *wxStockGDI::GetBrush(wxStockGDI::BRUSH_WHITE)*.

Stavolta abbiamo costruito il pennello esplicitamente, indicandone prima il colore, e poi la modalità di riempimento. Poiché la nostra è una applicazione di schizzi, ho scelto un riempimento a strisce diagonali, come quella mostrata in Figura 4. Potete sceglierne molte altre, comunque: dal colore solido, fino alla trama bitmap.

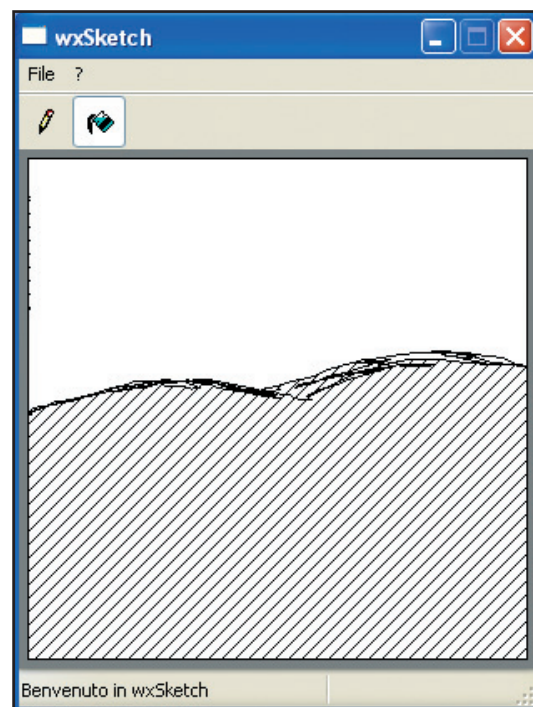



Fig. 4: ***Esempio di riempimento a diagonale rovesciata***

Per alterare la modalità di disegno dello strumento matita, invece, potremmo usare l'oggetto ***wxPen***, che è del tutto analogo a *wxBrush*. Può essere costruito a partire da una macro (come ***wxWHITE_PEN***, che sta per *wxStockGDI::GetBrush(wxStockGDI::PEN_WHITE)*), oppure via costruttore, indicando colore, dimensione del tratto, e modalità di disegno (solida, a trattini, a punti…).

ANDARE AVANTI

Lo spazio, purtroppo, è tiranno, è dobbiamo abbandonare la nostra applicazione di esempio in questo stadio embrionale. Portare a compimento altre funzionalità di *wxSketch* è probabilmente l'esercizio migliore che potrete mai trovare per migliorare la vostra comprensione dei meccanismi e delle possibilità di wxWidgets. Questi sono solo alcuni spunti la cui realizzazione offre grandi scoperte:

- Aggiungere altri strumenti (gomma, pennello, aerografo…)
- Per ogni strumento, prevedere una finestra di proprietà (dimensione tratto, colore…)
- Evitare condizioni if/else chilometriche per ogni strumento (usate una **classe Factory**)
- Permettere la scelta del colore da una tavolozza (usate ***wxColourDialog***)
- Permettere il salvataggio/caricamento su file (usate ***wxFileDialog***)
- Permettere la stampa del disegno (usate ***wxPrinter*** o ***wxPrintDialog***)
- Permettere l'undo/redo (trasformate ***Canvas::bitmap_*** in una lista e mantenete un iteratore)
- Far sì che le voci del menu Undo/Redo si disabilitino da sole quando non è possibile annullare o ripetere (usate ***wxUpdateUIEvent***)
- Implementare una funzionalità di zoom (usate il **command pattern** per dissociare la rappresentazione dal contenuto)
- Caricare gli strumenti dinamicamente via plug-in (usate **wxDynamicLibrary**)
- Inserire il tutto in un framework avanzato (usate **wxAui**, come visto nella scorsa puntata)

CONCLUSIONI

Ed eccoci giunti alla conclusione del nostro viaggio dentro wxWidgets, un framework tanto vasto che non sono bastate quattro puntate per scalfirne la superficie – spero, comunque, che questa carrellata sia servita a incuriosirvi.

Appuntamento al mese prossimo con altre mirabolanti avventure nel mondo del C++. Non mancate!

Roberto Allegra

L'AUTORE

Il sito www.robertoallegra.it contiene l'elenco degli articoli pubblicati in questa rubrica, con gli inevitabili approfondimenti ed errata corrige degli articoli e del codice. L'e-mail dell'autore è posta@robertoallegra.it.

MS VISUAL BASIC.NET PER MOBILE DEVICES

IMPARIAMO COME GESTIRE LE GALLERIE DI IMMAGINI ATTRAVERSO I CONTROLLI UTILIZZABILI PER LA PROGRAMMAZIONE DI DISPOSITIVI PORTATILI. COME ESEMPIO APPLICATIVO REALIZZEREMO UN'OROLOGIO PIUTTOSTO PARTICOLARE...

Nell'ultima puntata avete realizzato un orologio digitale che visualizza un'animazione sempre attiva, grazie a due oggetti Timer che lavorano in maniera asincrona: il primo scandisce l'ora con un grado di precisione basato sul secondo, l'altro aggiorna i frame ogni 2 centesimi di secondo.
In quest'articolo verranno introdotti ulteriori strumenti di programmazione, rendendo via via più complessa l'applicazione.

Fig. 1: L'applicazione Saturn mentre viene eseguita all'interno dell'emulatore per Pocket PC WM 5.0.

AFFINARE UN PROGRAMMA

Quando si crea un programma gli obiettivi da perseguire sono sia espliciti che impliciti. I primi riguardano il "che cosa" e il "come": che cosa deve fare l'applicazione e come deve apparire esteticamente, in altre parole back-end=motore e front-end=interfaccia grafica. I secondi sono più nascosti, ma altrettanto importanti poiché riguardano lo stile di programmazione, fondamentale per la comprensione del programma e la sua manutenzione successiva. Badate bene che i miglioramenti stilistici non coincidono sempre con una riduzione del codice, ma senza dubbio mirano a rendere più compatti gli algoritmi per una loro migliore leggibilità.

UTILIZZARE LA IMAGELIST

L'applicazione fino ad ora realizzata è perfettamente funzionante, ma ha almeno due difetti non trascurabili:
1. è necessario creare tante *PictureBox* quante sono le immagini che compongono l'animazione. Tenendo presente che, al fine di rendere fluida una sequenza animata, il numero di fotogrammi che compongono un file video sono solitamente 25 per secondo, è facilmente comprensibile che mano a mano che diventa più complesso e lungo il filmato diventa necessario cambiare approccio di programmazione.
2. L'istruzione *Select...Case* è sicuramente troppo pesante. Anche in questo caso il problema è generato dalla diretta dipendenza con il numero delle immagini: i rami *Case* della *Select* sono tanti quanti i fotogrammi che si vogliono visualizzare. A lungo andare
Un rimedio efficace è utilizzare l'oggetto *ImageList*, presente nella Casella degli strumenti, al posto di tutte le *PictureBox* a parte quella battezzata *picGC*. Una volta inserito nella form corrente è necessario innanzitutto modificarne la proprietà *Name* con *imaGC* e la proprietà *ImageSize* che deve contenere le

dimensioni dei fotogrammi: *Width=116* e *Height=214*.

Fig. 2: La finestra di dialogo della proprietà Images consente d'introdurre tutte le immagini necessarie nella ImageList.

Successivamente inserite le varie immagini facendo clic sul pulsantino della proprietà *Images* a destra di (*Collection*). A questo punto non dovete fare altro che modificare il contenuto dell'evento *Tick* del *Timer tmrAnima*, che è deputato alla visualizzazione dell'animazione. Prima, infatti, eravate costretti ad utilizzare una *Select...Case* per verificare il valore corrente del contatore *ctr* e, di conseguenza, visualizzare l'immagine corretta in *picGC.Image*. Ora, invece, è sufficiente far puntare la *PictureBox* al metodo *Item* della *ImageList*, indicizzato con il valore attuale di *ctr*. Il nuovo evento *Tick* è dunque il seguente:

```
Private Sub tmrAnima_Tick(...) Handles
                                   tmrAnima.Tick
    Static ctr As Integer = 0
    ctr += 1
    picGC.Image.Dispose()
    picGC.Image = imaGC.Images.Item(ctr - 1)
    If ctr = 9 Then ctr = 0
End Sub
```

Notare, infine, che è stato necessario introdurre l'istruzione picGC.Image.Dispose() per rilasciare ogni volta le risorse utilizzate da picGC ed evitare un errore di out of memory.

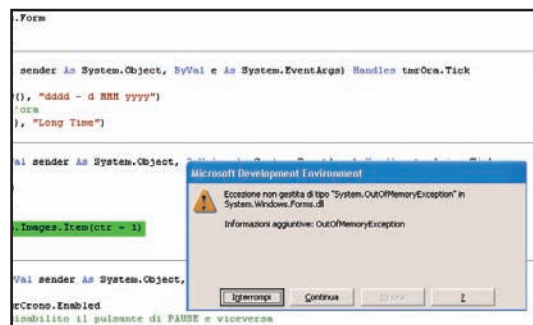



Fig. 3: Il metodo Dispose() applicato alla PictureBox evita un brutto errore di out of memory che qui visualizziamo. Se ricevete questa eccezione, dopo avere corretto il codice, è consigliabile riavviare il Pocket PC o l'emulatore per evitare d'incappare in strani errori che vi possono portare fuori strada.

UN'ANIMAZIONE PIÙ COMPLESSA

Ora che sapete come ottimizzare il codice, potete rendere più attraente l'orologio introducendo un maggior numero d'immagini. Inserite, dunque, nella ImageList le 35 immagini che riguardano Saturno e date nomi più generici alla *PictureBox* e alla *ImageList*, rispettivamente *picAnima* e *imaAnima*. Facciamo, poi, le dovute correzioni all'evento *Tick* di *tmrAnima* che consistono essenzialmente nell'introduzione delle seguenti istruzioni:

```
picAnima.Image.Dispose()
picAnima.Image = imaAnima.Images.Item(ctr - 1)
If ctr = 34 Then ctr = 0
```

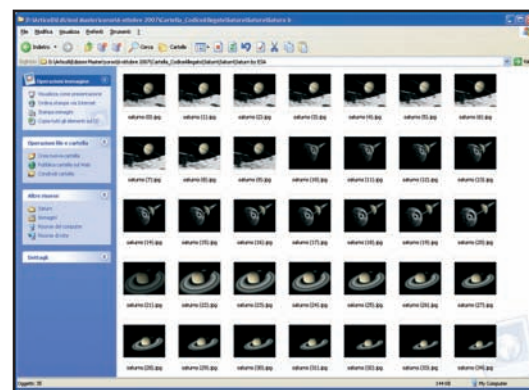



Fig. 4: Le immagini fornite dall'ESA che rendono più accattivante l'applicazione.

DALL'OROLOGIO AL CRONOGRAFO

Il prossimo passo è quello di trasformare il nostro orologio in un cronografo, aggiungendo le funzioni proprie di un cronometro.
Visualizzate la Toolbox e fate doppio clic sull'oggetto *Label*, per riportarlo all'interno della form. Dopo avere fatto clic sulla label e avere premuto F4, impostate alcune proprietà come segue:

```
(Name)=lblOre
Font.Size=20
Font.Bold=True
ForeColor=RoyalBlue
Location.X=30
Location.Y=230
```

Poiché l'obiettivo è quello di creare un crono-

IMPLEMENTAZIONE DELLA IF

Quando si vuole testare un'ulteriore condizione è possibile aggiungere uno o più rami ElseIf all'interno di una If. La sintassi viene riportata di seguito.

```
If <condizione> Then
    [istruzioni]
ElseIf <altra condizione> Then
    [istruzioni]
+ <altre eventuali elseif>
Else
    [istruzioni]
End If
```

Si ricorda che le clausole ElseIf ed Else sono entrambe facoltative.

metro che visualizza ore, minuti, secondi e centesimi di secondo, avete bisogno di altre 3 label. Le creerete per "clonazione" dalla prima. Selezionate, dunque, *lblOre* e dal menu *Edit* scegliete prima *Copy* e poi *Paste*. Ripetete la stessa operazione altre due volte, in maniera tale da avere quattro controlli label che, a parte nome e posizione, avranno tutti le stesse proprietà di *lblOre*. Infine battezzateli, definite i colori e localizzateli correttamente sullo schermo rispettivamente come indicato di seguito:

(Name)=lblMin
ForeColor= DeepSkyBlue
Location.X=76
Location.Y=230

(Name)=lblSec
ForeColor= MediumSpringGreen
Location.X=120
Location.Y=230

(Name)=lblCen
ForeColor= Aquamarine
Location.X=166
Location.Y=230

Tutte e quattro le label avranno anche le seguenti impostazioni relative alla grandezza del font, alle dimensioni della casella e alla centratura del testo:

Font.Size=20
Size.Width=44
Size.Heigth=32
TextAlign=TopCenter

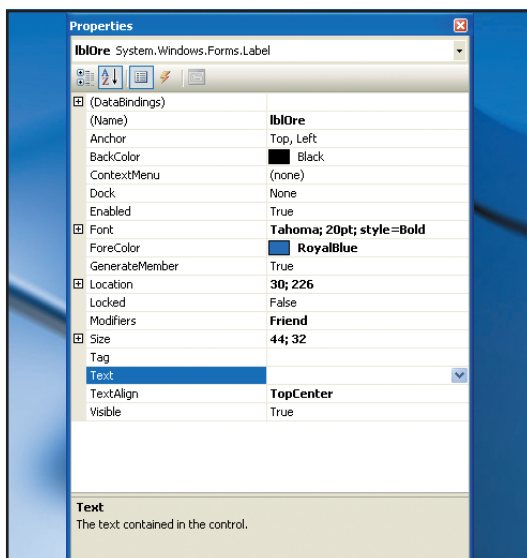



***Fig. 5:** Le proprietà delle Label lblOre, lblMin, lblSec e lblCent sono importanti al fine di visualizzare correttamente il cronometro.*

Ora serve un pulsante che avrà la duplice funzione di fare partire il cronometro e di stopparlo. Prelevate un controllo *Button* dalla Toolbox e definitene le seguenti proprietà:

(Name)=btnCrono
Location.X=0
Location.Y=241
Size.Width=32
Size.Heigth=16
Text=Crono

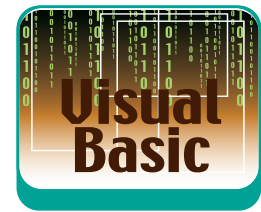
L'ultimo pulsante che aggiungerete serve a mettere in pausa il cronometro.

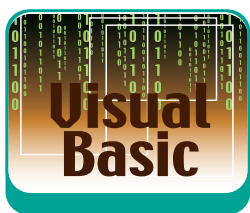
(Name)=btnCronoPausa
Location.X=208
Location.Y=241
Size.Width=32
Size.Heigth=16
Text=Pausa

Infine inserite un altro oggetto *Timer* dalla Toolbox. Le proprietà da modificare sono (Name) da uguagliare a *tmrCrono* e *Interval* che deve essere impostato a 10 perché il cronometro deve "girare" ogni centesimo di secondo.

GLI ALGORITMI DEI PULSANTI

Passiamo ora al back-end, ossia alla costruzione di quegli algoritmi che consentono al cronografo di funzionare realmente. Fate doppio clic sul pulsante *btnCrono* per "entrare" automaticamente nel suo evento *Click*. Qui dovete agire sulla proprietà *Enabled* del *Timer* per abilitarlo o disabilitarlo ad ogni pressione successiva: questo ne consentirà l'attivazione in fase run time. Inoltre se *tmrCrono.Enabled* è uguale a *False* disabilitate anche il pulsante di *Pausa* e viceversa. Infine è importante notare che quando *tmrCrono* è uguale a False le variabili *hr, mi, se* e *ce*, che come vedremo sono alla base del funzionamento del cronometro, devono essere uguagliate a zero proprio per azzerare il cronometro.

```
Private Sub btnCrono_Click(...) Handles
                                btnCrono.Click
    tmrCrono.Enabled = Not tmrCrono.Enabled
    btnCronoPausa.Enabled = tmrCrono.Enabled
    If tmrCrono.Enabled Then
        btnCrono.Text = "Stop"
    Else
        lblOre.Text = "" : lblMin.Text = ""
        lblSec.Text = "" : lblCen.Text = ""
        hr = 0 : mi = 0 : se = 0 : ce = 0
```

ROUTINE PERSONALIZZATE

Sub e Function sono routine che possono essere create dal programmatore o fornite da VB.NET. Un esempio delle seconde è la funzione MsgBox() che consente di visualizzare un messaggio. Da ricordare che le Sub non restituiscono valori in uscita, mentre le Function sì.

NOTA

LE IMMAGINI DELL'ESA

Si ringrazia l'Agenzia Spaziale Europea (http://www.esa.int) per le immagini che hanno consentito la realizzazione dell'animazione. Si tratta di ben 35 immagini estratte da un file WMV dell'ESA, che rappresentano il pianeta Saturno mentre esegue la propria rotazione all'interno del Sistema Solare.

```
            btnCrono.Text = "Crono"
        End If
    End Sub
```

Anche l'evento *Click* del pulsante *btnCronoPausa* ovviamente ad ogni pressione abilita o disabilita il timer *tmrCrono*, ma abilita o disabilita anche il pulsante di attivazione o spegnimento del cronometro. In pausa, infatti, non avrebbe senso fare clic su *btnCrono*.

```
    Private Sub btnCronoPausa_Click(...) Handles
                                    btnCronoPausa.Click
        tmrCrono.Enabled = Not tmrCrono.Enabled
        btnCrono.Enabled = tmrCrono.Enabled
    End Sub
```

IL MOTORE DEL CRONOMETRO

Il motore vero proprio è costituito dal *Timer tmrCrono* e dalla procedura *procIncr* richiamata al suo interno. Il codice contenuto nel *Timer* fa sì che il cronometro possa partire, incrementando centesimi, secondi, minuti e ore correttamente. La sua attivazione avviene aggiungendo un'unità ai centesimi di secondo, ossia con *ce+=1*. Successivamente viene richiamata tre volte la routine *procIncr* per aggiornare le *label* che compongono il cronometro.

```
    Private Sub tmrCrono_Tick(...) Handles
                                    tmrCrono.Tick
        ce += 1
        procIncr(ce, se, lblCen.Text, lblSec.Text, True)
        procIncr(se, mi, lblSec.Text, lblMin.Text, False)
        procIncr(mi, hr, lblMin.Text, lblOre.Text, False)
        If Len(hr.ToString) = 1 Then
            lblOre.Text = "0" & hr.ToString
        End If
        If hr = 24 Then
            ce = 0 : se = 0 : mi = 0 : hr = 0
        End If
    End Sub
```

La routine *procIncr* è estremamente compatta e deve essere analizzata con il debug per essere compresa nel dettaglio. Si consiglia, quindi, d'impostare un punto d'interruzione con F9 sulla prima riga al di sotto del suo nome e, poi, di seguirla passo dopo passo. Tenete conto, comunque, che essendo basata su parametri (quelli tra parentesi) e su una *If*, consente di incrementare tutti i valori del cronometro.
E' stato, poi, utilizzato il metodo *ToString* perché si vuole convertire un valore numerico in carattere, per visualizzarlo nella *Label* (rappresentata dai parametri *xLab1* e *xLab*).

```
    Private Sub procIncr(ByRef xVar As Integer, _
```

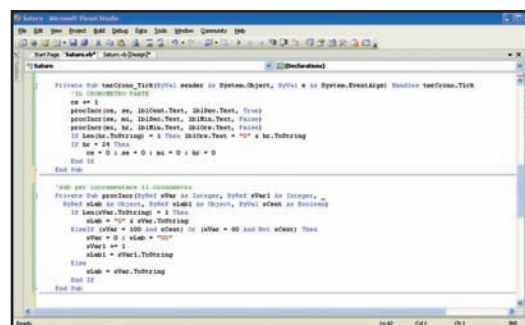



Fig. 6: L'algoritmo che permette il funzionamento del cronometro può essere analizzato nel dettaglio con l'ausilio del debug (impostare i punti d'interruzione con il tasto F9).

```
            ByRef xVar1 As Integer, ByRef xLab As
                                              Object, _
            ByRef xLab1 As Object, ByVal xCent As
                                              Boolean)
        If Len(xVar.ToString) = 1 Then
            xLab = "0" & xVar.ToString
        ElseIf (xVar = 100 And xCent) Or _
                (xVar = 60 And Not xCent) Then
            xVar = 0 : xLab = "00"
            xVar1 += 1
            xLab1 = xVar1.ToString
        Else
            xLab = xVar.ToString
        End If
    End Sub
```

Sono stati, infine, introdotti due pulsanti di comando: *btnEsci* e *btnFerma*. Il primo, grazie a all'istruzione *Application.Exit()*, consente di chiudere l'applicazione senza lasciarla appesa nella memoria del Pocket PC. Il secondo blocca (*Ferma*) o riattiva (*OK*) l'animazione, agendo su *tmrAnima.Enabled*.

```
    Private Sub btnFerma_Click(...) Handles
                                    btnFerma.Click
        tmrAnima.Enabled = Not tmrAnima.Enabled
        If tmrAnima.Enabled Then
            btnFerma.Text = "Ferma"
        Else
            btnFerma.Text = "OK"
        End If
    End Sub
```

GLI ULTIMI ACCORGIMENTI

Per quanto possa sembrare una banalità, la creazione dell'icona del progetto non deve essere trascurata. L'efficacia di un programma e la sua capacità di avere successo dipende da numerosi fattori

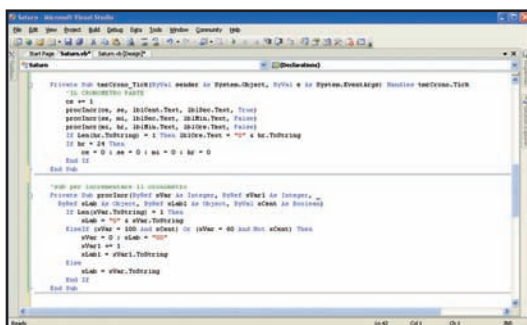



Fig. 7: L'applicazione in esecuzione all'interno dell'emulatore per Windows Mobile 2003 Second Edition.

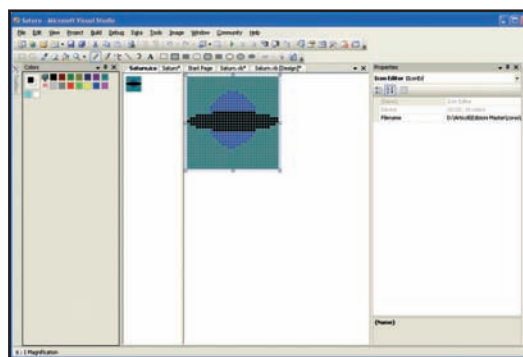
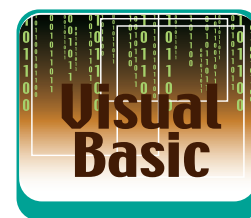



Fig. 8: L'editor delle icone di VB.NET.

e cioè: elevate prestazioni, semplicità d'uso, utilità delle funzionalità rispetto alle esigenze degli utenti, riempimento dei tempi di attesa (un'applicazione che deve fare pesanti elaborazioni deve visualizzare qualcosa nel frattempo) e ultima, ma non per importanza, interfaccia grafica. Quando si fornisce un programma la prima cosa che viene notata è l'icona, dunque impariamo a crearla e ad impostarla. Fate clic destro nel Solution Explorer sul nome del progetto (Saturn), scegliete *Add-New Item* e scorrete la progress bar a destra per selezionare con un solo clic *Icon File*. Nel campo *Name*, al posto di *Icon1.ico*, digitate il nome che volete dare all'icona, ad esempio *Saturn.ico* e fate clic sul pulsante *Add*. A questo punto appare l'ambiente di lavoro che consente di creare l'icona del progetto e che è composto dai seguenti elementi: la barra con la tavolozza dei colori, due riquadri che contengono rispettivamente l'icona in dimensioni reali e l'icona ingrandita (si può disegnare su entrambi), il menu Image con i suoi elementi e la barra delle icone Icon Editor. Le impostazioni di default dell'icona sono 32x32 e 16 colori, ma sono modificabili selezionando il menu *Image-New Image Type* e il nuovo formato presente nella finestra di dialogo oppure il menu *Image-Current Icon Image Types*, mantenendo lo stesso formato (es.32x32), ma modificando il numero di colori supportati.
Una volta definito il formato dell'icona, potete crearla sia disegnandola a mano libera sia copiando un disegno o parte di esso nell'apposito riquadro. Gli strumenti disponibili sono numerosi e simili a quelli che vengono solitamente messi a disposizione da altri programmi di editing d'immagini: matita, pennello, aerografo, selezioni, figure geometriche, ecc. Tramite il menu Image, inoltre, è possibile invertire i colori, capovolgere l'immagine orizzontalmente o verticalmente oppure ruotarla di 90 gradi. Per ridurre i tempi di lavoro copiate l'immagine da una già esistente ricordando, però, che deve avere le stesse dimensioni dell'area dell'icona; di solito è necessario ridimensionarla ad hoc. Una volta ultimato il disegno ricordate di definire "trasparenti" le aree esterno ad esso, in questo modo lo sfondo dell'immagine è esattamente quello del desktop del Pocket PC e l'icona è molto piú professionale. Dovete, quindi, selezionare lo strumento aerografo e, poi, sulla tavolozza fare clic sul piccolo monitor di colore verde. A questo punto fate clic sulle aree del disegno che volete rendere trasparenti e il gioco è fatto. Dovete ora salvare l'icona e chiudere l'editor. Tenete presente che l'icona, una volta creata con il tool di .NET, genera automaticamente un file .ico nella directory principale del progetto. Ciò significa che se per altri motivi avete bisogno di generare un file .ico lo potete fare utilizzando l'editor di icone di Visual Basic.NET. Per concludere il vostro sforzo dovete abbinare l'icona al progetto. Scegliete *Project-Saturn* Properties, poi la scheda *Application*, fate clic sulla casella combinata *Icon* e selezionate Saturn.ico.

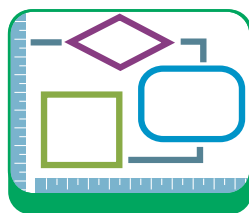
Enrico Bottari

Fig. 8: L'applicazione in esecuzione all'interno dell'emulatore per Windows Mobile 2003 (questa volta prima edizione).

NOTA

UTILIZZARE IL SOFTWARE ALLEGATO

Come prima cosa copiate l'intero progetto Saturn dal CD al vostro PC. Per avviarlo all'interno di Visual Basic.NET entrate nella cartella Saturn e fate doppio clic sul file Saturn.sln.

GESTISCI GLI ALBERI CON IL COMPOSITE

QUANDO SI PROGRAMMA È FACILE IMBATTERSI IN STRUTTURE RICORSIVE. CON UN QUALSIASI LINGUAGGIO OBJECT-ORIENTED E IL PATTERN COMPOSITE POTETE TRASFORMARE ANCHE GLI ALBERI PIÙ CONTORTI IN SEMPLICI RELAZIONI TRA OGGETTI

Chattoo, il nuovo progetto a cui state lavorando, ha tutte le caratteristiche delle migliori applicazioni Web 2.0: interfaccia AJAX, grafica tondeggiante, e soprattutto un nome buffo che contiene due "o" di seguito. Beatrice ha appena iniziato a lavorarci, ma sa che ben presto potrà mostrare il primo prototipo a stuoli di finanziatori ansiosi di aprire il portafogli.
Il progetto di Beatrice è un network sociale dove ciascun utente è un "contatto". Chiunque può assegnare dei "tag" (cioè etichette) a qualsiasi contatto. Ecco la classe Java che implementa un contatto:

```
import java.util.HashSet;
import java.util.Set;

public abstract class Contact {
    private String name;
    private final Set tags = new HashSet();

    public Contact(String name) {
        this.name = name;
    }

    public void addTag(String tag) {
        tags.add(tag);
    }

    public Set getTags() {
        return tags;
    }
    public String toString() {
        return name + " " + getTags().toString();
    }
}
```

Un contatto ha un nome e un insieme di tag. Il nome viene passato direttamente al costruttore, mentre i tag possono essere aggiunti in qualsiasi momento. Contact conserva i tag in un Set, una collezione che elimina automaticamente gli eventuali duplicati. La classe ha anche un metodo toString() che restituisce nome e tag formattati come una stringa. Contact è una classe astratta, perché i contatti appartengono in realtà ad una tra due possibili sottoclassi. Il tipo più semplice di contatto è un "utente":

```
public class User extends Contact {
    public User(String name) {
        super(name);
    }
}
```

In questo primo prototipo, uno User non ha alcuna caratteristica propria a parte quelle ereditate da Contact. In futuro, Beatrice arricchirà questa classe con altre funzionalità. Nel frattempo si può già scrivere del codice che crea un utente e gli assegna dei tag:

```
User u = new User("Pierah");
u.addTag("cubista");
u.addTag("vacanze");
u.addTag("2007");
System.out.println(u);
```

Il risultato è:

```
Pierah [cubista, vacanze, 2007]
```

Il secondo tipo di contatto sono i "gruppi", che sono collezioni di utenti:

```
import java.util.LinkedList;
```

COME LEGGERE QUESTO ARTICOLO

In questo articolo usiamo alcune funzionalità avanzate del linguaggio Ruby, che non tutti conoscono. Non preoccupatevi: non dovete capire ogni singola riga, e quando vedete che qualche dettaglio non viene spiegato, questo significa che non è importante. Immaginate di guardare il codice da sopra la spalla di Beatrice, che vi spiega a grandi linee come funziona. La cosa importante è che capiate il principio su cui si basa il pattern Command.

```
import java.util.List;

public class Group extends Contact {

    private List children = new LinkedList();

    public Group(String name) {
        super(name);
    }

    public void addChild(User member) {
        getChildren().add(member);
    }

    public List getChildren() {
        return children;
    }
}
```

Ciascun gruppo chiama i suoi utenti “figli” (“children”), per motivi che vedremo più avanti. Il gruppo conserva i propri figli in una lista. *Group* eredita da *Contact*, quindi è possibile assegnare dei tag ai gruppi come agli utenti.
Il sistema di Beatrice può mostrare su una pagina web tutti i tag associati ad un gruppo. Per farlo, deve raccogliere i tag del gruppo e quelli degli utenti che appartengono al gruppo:

```
public class Chattoo...
    public static void main(String[] args) {
         Group group = new Group("Club dell'Alce");
        User u = new User("Jyino");
        u.addTag("ingegnere");
        group.addChild(u);
        group.addChild(new User("Pyera"));
        group.addTag("alce");

        System.out.println(allTags(group));
    }
```

In questo test, il gruppo “Club dell'Alce” ha un tag (“alce”) e contiene due utenti, uno dei quali ha a sua volta un tag (“ingegnere”). Il metodo allTags() crea un insieme e gli aggiunge tutti i tag del gruppo. Poi fa un ciclo su tutti gli User “figli” del gruppo e aggiunge tutti i tag di ciascuno User all'insieme:

```
public class Chattoo...
    private static Set allTags(Group group) {
        Set result = new HashSet();
        result.addAll(group.getTags());
        for (Iterator i = group.getChildren().iterator();
                                              i.hasNext();)
            result.addAll(((User) i.next()).getTags());
        return result;
    }
```

Il risultato del test è:

```
[ingegnere, alce]
```

PICCOLE VITTORIE

Beatrice chiede agli amici di provare il prototipo del suo network sociale, ma bastano pochi minuti perché salti fuori il primo utente insoddisfatto. Uno dei tester vorrebbe poter inserire dei gruppi in un altro gruppo. In questo modo sarebbe possibile costruire una gerarchia di gruppi, dove ciascun gruppo può contenere sia utenti che altri gruppi. Per fortuna la modifica è molto semplice: dato che Group e User ereditano entrambe da Contact, basta fare in modo che i figli di un gruppo possano essere dei generici Contact:

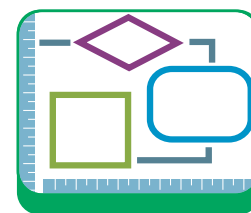
```
public class Group extends Contact...
    public void addChild(Contact member) {
        getChildren().add(member);
    }
```

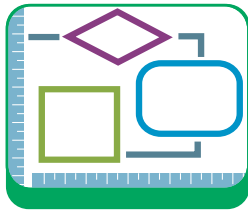
Ora un gruppo può contenere sia utenti che altri gruppi. Quindi possiamo avere tutti i gruppi annidati che vogliamo. La struttura risultante è un gruppo che contiene utenti e gruppi, che a loro volta contengono utenti e gruppi. Questo è quello che in informatica si chiama un “albero”. Il gruppo più esterno è la “radice”, e ciascun gruppo o utente è un “nodo”. Dato che gli utenti non possono avere figli, un utente è un nodo speciale che si chiama un “foglia”. Ad esempio:

```
public class Chattoo...
    public static void main(String[] args) {
        Group group = new Group("Club dell'Alce");
        User u = new User("Jyino");
        u.addTag("ingegnere");
        group.addChild(u);
        group.addChild(new User("Pyera"));
        group.addTag("alce");
        group.addTag("bologna");

        Group subgroup = new Group("Giovani
                                            Marmotte");
        subgroup.addChild(new User("Franko"));
        subgroup.addTag("marmotte");
        group.addChild(subgroup);

        System.out.println(allTags(group));
    }
```

Il gruppo “Club dell'Alce” ha ora due utenti e un sottogruppo che a sua volta contiene un utente. Ma quando Beatrice lancia questo programma, il metodo *allTags()* esplode con una *ClassCastEx-*

UNA PICCOLA SFIDA

Se conoscete un po' di Ruby, potete provare a completare questo articolo aggiungendo una funzionalità di Redo che riapplica in sequenza le operazioni annullate. È un esercizio che vi richiederà qualche minuto, ma vi darà anche soddisfazione.

ception. Il problema è che *allTags()* è convinto che i figli di un gruppo siano sempre degli *User*, e infatti contiene un cast esplicito alla classe *User*. La nuova versione dei gruppi, invece permette di avere figli di classe *Group*.
Poco male, pensa Beatrice. Basterà qualche piccola modifica:

```
public class Chattoo...
    private static Set allTags(Contact c) {
        Set result = new HashSet();
        result.addAll(c.getTags());
        if (c instanceof User)
            return result;
        else if (c instanceof Group) {
            for (Iterator i =
            c.getChildren().iterator(); i.hasNext();)
                result.addAll(allTags((Contact)
                                        i.next()));
            return result;
        }
        throw new RuntimeException("Tipo di
                                contatto sconosciuto");
    }
```

Questa versione di *allTags()* è molto più complicata dalla precedente. Per cominciare, accetta qualsiasi contatto – sia un *Group* che uno *User*. Il metodo aggiunge i tag del contatto ad un insieme ed esamina il tipo del contatto. Se scopre di trovarsi di fronte ad uno *User*, restituisce semplicemente i tag che ha appena estratto. Se si trova di fronte ad un *Group*, chiama ricorsivamente sé stesso, aggiungendo all'insieme dei risultati i tag di ciascun "figlio". Se il contatto attuale non è né un utente né un gruppo, il metodo lancia un'eccezione.
Questo codice funziona, ma Beatrice è disgustata dal solo pensiero di lasciarlo nella sua applicazione. Se un giorno dovessero nascere altre sottoclassi di *Contact* oltre a *Group* e *User*, questo codice dovrebbe essere modificato. Basterebbe dimenticarsene per essere tempestati da antipatiche eccezioni. Il modo giusto per risolvere il problema, ragiona Beatrice, è usare il polimorfismo. Tutti i metodi che servono ad *allTags()* dovrebbero essere disponibili su tutti i possibili *Contact*.
Guardando il codice di *allTags()*, Beatrice nota che l'unico metodo che crea problemi è *getChildren()*, che esiste su *Group* ma non su *User*. Niente impedisce però di aggiungerlo anche a *User*, a patto che l'elenco dei figli di uno *User* sia sempre vuoto:

```
import java.util.LinkedList;
import java.util.List;

public class User extends Contact {

    public User(String name) {
        super(name);
    }

    public List getChildren() {
        return new LinkedList();
    }
}
```

Perché *getChildren()* possa essere chiamato attraverso la classe *Contact*, dobbiamo definirlo anche lì. Dato che il metodo si comporta in modo diverso in ciascuna sottoclasse concreta di *Contact*, tanto vale dichiararlo astratto:

```
public abstract class Contact...
    public abstract List getChildren();
```

Ora il codice di allTags() diventa semplice:

```
public class Chattoo...
    private static Set allTags(Contact c) {
        Set result = new HashSet();
        result.addAll(c.getTags());
        for (Iterator i = c.getChildren().iterator();
                                        i.hasNext();)
            result.addAll(allTags((Contact) i.next()));
        return result;
    }
```

allTags() aggiunge al risultato sia i tag del contatto che gli viene passato, che (ricorsivamente) i tag di tutti i suoi figli. La ricorsione termina negli *User*, che non hanno figli. Ecco il risultato del test:

```
[bologna, ingegnere, alce, marmotte]
```

Beatrice è quasi soddisfatta, ma il suo occhio esperto identifica un'altra possibile miglioria.

SPOSTANDO CODICE

Il metodo *allTags()* è migliorato, ma è ancora bruttino. Questo metodo esplora l'intero albero dei gruppi ed estrae i tag da ciascun nodo. Nella programmazione a oggetti, è spesso una buona idea che il proprietario dei dati sia anche il responsabile delle relative operazioni. Visto che i tag appartengono ai contatti, sarebbe bene spostare l'estrazione dei tag sulla classe *Contact*. Beatrice cancella il metodo *allTags()* dal programma principale, e lo trasforma in un metodo *getTags()* su *Contact*:

```
public class Chattoo {
    public static void main(String[] args) {
        Group group = ...

        Group subgroup = ...

        System.out.println(group.getTags());
    }
}
```

Dato che l'implementazione di *getTags()* dipende dal fatto che il contatto sia un utente o un gruppo, Beatrice definisce *Contact.getTags()* come un metodo astratto – e visto che c'è, fa lo stesso con *addTag()*:

```
public abstract class Contact...
    public abstract void addTag(String tag);
    public abstract Set getTags();
```

La gestione dei tag viene trasferita negli utenti:

```
public class User extends Contact...
    private Set tags = new HashSet();

    public void addTag(String tag) {
        tags .add(tag);
    }

    public Set getTags() {
        return tags;
    }
```

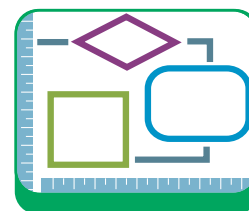
I gruppi, invece, si limitano a delegare le due operazioni ai proprio componenti. Per aggiungere un tag, un gruppo lo aggiunge semplicemente a tutti i figli. Per restituire l'elenco dei tag, il gruppo raccoglie in un insieme i tag di tutti i figli:

```
public class Group extends Contact...
    public void addTag(String tag) {
        for (Iterator i =
           getChildren().iterator(); i.hasNext();)

            ((Contact) i.next()).addTag(tag);
    }

    public Set getTags() {
        Set result = new HashSet();
        for (Iterator i =
           getChildren().iterator(); i.hasNext();)

            result.addAll(((Contact)
                                    i.next()).getTags());
        return result;
    }
```

Beatrice sa che questa nuova versione del sistema non è esattamente identica alla precedente. Nella versione precedente, i tag di un gruppo erano indipendenti da quelli degli utenti. Ora, invece, i gruppi non sono più "taggabili" per conto proprio. Se assegno un tag ad un gruppo, non faccio altro che assegnarlo a tutti gli utenti del gruppo (e ricorsivamente a tutti gli utenti dei suoi sotto-gruppi). Questo significa che se aggiungo un utente al gruppo dopo aver aggiunto il tag, il nuovo utente non avrà il tag. Dopo averci pensato un po' su, Beatrice decide che questo nuovo comportamento le piace di più: i gruppi devono essere solo contenitori, e sono gli utenti che "posseggono" i tag. Il primo modello del sistema va bene così, e Beatrice può andare a dormire soddisfatta.
Per noi, invece, è il momento di dare uno sguardo di insieme al sistema.

IL PATTERN COMPOSITE

Diamo un'occhiata alle classi. Il client parla ad un'interfaccia *Contact*, che definisce operazioni per gestire i tag e per leggere l'elenco dei "figli". Esistono due classi concrete che implementano *Contact*. Una, *User*, è un semplice utente. L'altra, *Group*, contiene una collezione di *Contact*, ciascuno dei quali può essere a sua volta uno *User* o un *Group*. Entrambe le classi implementano le operazioni definite da *Contact*, e *Group* ha anche un'operazione in più per aggiungere elementi alla sua collezione di figli.
Il risultato di questo semplice schema è una struttura ad albero simile ad un file system. Un file system è composto da file e directory, e ciascuna directory può contenere file e altre directory. I file non possono contenere altri file,

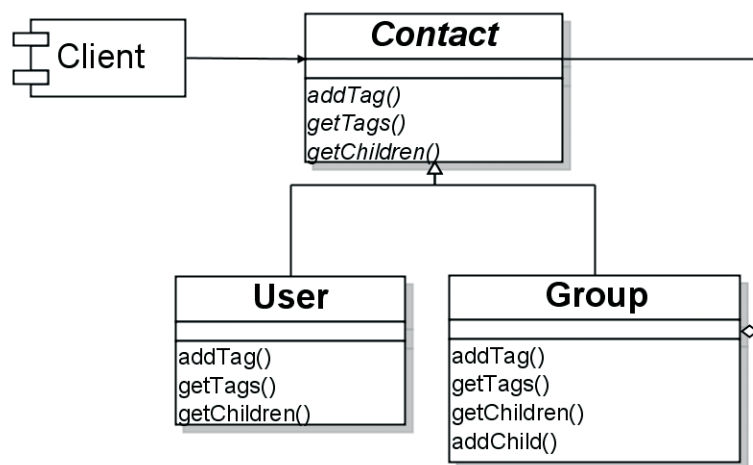
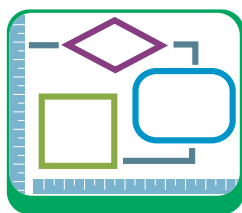
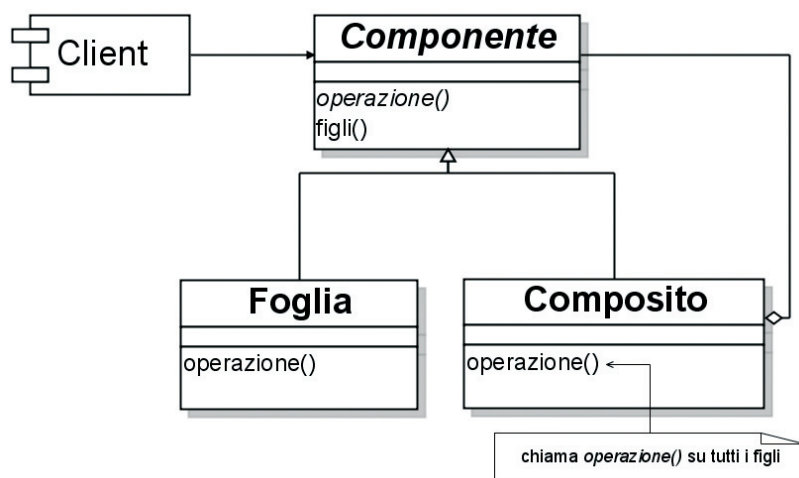



Figura 1: Le classi del progetto

quindi sono "foglie". Nel nostro caso, i gruppi fanno la parte delle directory e gli utenti quella dei file.
Questo tipo di struttura gerarchica è comunissima. Esiste un pattern apposta per gestirlo, di nome Composite. Basta cambiare i nomi nel diagramma per avere una descrizione generale del Composite:

Un albero è composto da normali nodi, che possono avere dei figli, e "foglie", che sono nodi senza figli. L'idea alla base del pattern è quella di usare due classi diverse per indicare un normale nodo ("Composito") e una foglia ("Foglia"). Entrambe le classi hanno un supertipo comune ("Componente"), che in Java può essere una classe, probabilmente astratta, o un'interfaccia.

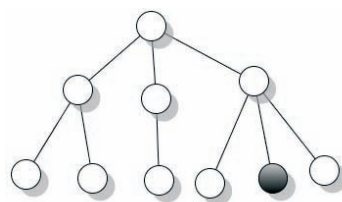
Normalmente il client non ha bisogno di sapere se il componente con cui sta parlando è una Foglia o un Composito: tutto quello che gli serve è sapere che ha a che fare con un componente, in modo da poter chiamare metodi come *operazione()*. Il cuore del pattern sta nel fatto che ciascun Composito conserva una lista di Componenti, che quindi a loro volta possono essere Foglie o altri Compositi.
Mentre le Foglie implementano l'*operazione()*, il Composito la delega di solito ai propri Componenti. Il risultato è una versione "ad oggetti" del classico algoritmo ricorsivo che si usa per applicare un'operazione a tutte le foglie di un albero:

a – Parti dal nodo di livello più alto (la "radice").
b – Se il nodo corrente è una foglia, chiama l'*operazione()*.
c – Se il nodo corrente non è una foglia, esegui (b) e (c) su ciascun figlio del nodo.

Come tutti i pattern, anche Composite può avere molte varianti. Ad esempio, le operazioni per aggiungere o rimuovere figli possono v ivere nel Composito o nel Componente.
Nel sistema di Beatrice, l'operazione *addChild()* vive nella classe *Group*. Se l'operazione fosse stata disponibile sulla superclasse *Contact*, infatti, qualcuno avrebbe potuto chiamarla su uno *User* e aggiungere dei figli alle foglie dell'albero. Per evitare questa eventualità, avremmo probabilmente dovuto fare l'override di *addChild()* in *User* e lanciare un'eccezione.

GIOIE E DELIZIE DELLA RICORSIONE

Le funzioni e i metodi "ricorsivi" (cioè che chiamano sé stessi) sono uno scoglio arduo per molti programmatori principianti. Qualcuno dice che esistono due tipi di programmatori: quelli che capiscono al volo la ricorsione, e quelli che non la capiranno mai. Se pensate di essere nella seconda categoria, non disperate. La ricorsione è molto semplice, una volta afferrato il principio di base.

L'esempio che si usa tipicamente per introdurre la ricorsione è il calcolo del fattoriale, ma esistono esempi più semplici. Immaginate un albero dove ciascun nodo può essere bianco o nero. Quasi tutti i nodi sono bianchi, ma ogni tanto ci si imbatte in un nodo nero. Disegnate uno di questi alberi su un foglio di carta:

Ora immaginate di dover scrivere un'operazione di nome cercaNodiBianchi che prende un nodo (inizialmente la "radice" dell'albero), e restituisce "vero" se l'albero contiene un nodo nero e "falso" in caso contrario. Come fareste voi, se foste un algoritmo, a scovare i nodi neri? Provate a scrivere su un foglio i passi dell'operazione, e vedrete che la ricorsione non tarderà a saltare fuori.

CONCLUSIONI

Composite è un pattern da considerare ogni qual volta ci troviamo di fronte ad una struttura gerarchica. Un esempio che abbiamo già considerato è quello del file system. Un altro esempio sono le figure in un programma di disegno vettoriale.
Ciascuna figura può essere modificata per conto proprio, ad esempio per cambiarne le dimensioni o lo spessore delle linee. E' anche possibile creare gruppi di figure che vengono modificate insieme.
Un gruppo può includere delle singole figure, o altri gruppi. In generale, se guardate in qualsiasi sistema complesso vi troverete quasi certamente qualche struttura a forma di albero.

Paolo Perrotta

SOFTWARE SUL CD

Realbasic 2007

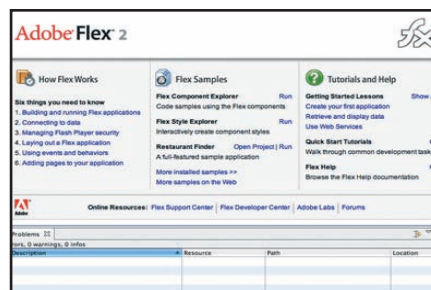
IL MULTIPIATTAFORMA CHE FUNZIONA

Il sogno di molti è avere a disposizione un ambiente semplice, potente e RAD che produca eseguibili per le piattaforme Windows, Macintosh, Linux. Questo sogno oggi è realtà e si concretizza in un unico nome: Realbasic 2007. Realbasic è un prodotto straordinario. Un ambiente RAD disponibile su tutte le piattaforme che fa da cappello ad un compilatore altrettanto multipiattaforma. Il linguaggio che fa da sfondo a tutto questo è un Basic avanzato, ad oggetti ovviamente. Abbiamo provato questa versione 2007 sulle nostre macchine e ne siamo rimasti grandemente impressionati, per la velocità, la semplicità e la completezza dell'ambiente. Inoltre Real Basic è quasi completamente compatibile con il vecchio codice Visual Basic. Questo vi consente di fare diventare le vostre applicazioni VB multipiattaforma in modo estremamente semplice e veloce

Directory:REALbasicSetup.exe

FLEXBUILDER 2.0

IL FRAMEWORK PER LE APPLICAZIONI FLASH

Straordinario! Non ci sono parole per descrivere questo nuovo prodotto della linea Adobe che in un sol colpo fa piazza pulita di un vecchio modo di pensare alle applicazioni per inventarne uno del tutto nuovo, semplice ed estremamente potente. Chi ha letto questo numero di ioProgrammo sa già come si scrive un programma Flex. Si prepara un file XML, lo si da in pasto al compilatore e si ottiene in uscita un file SWF ovvero utilizzabilie dal noto player flash di Macromedia. Quali sono i vantaggi? Ce ne sono diversi, i più

facili da intuire sono la completa indipenza dal sistema operativo senza per questo doversi affidare alla macchinosità di java. La possibilità che la stessa applicazione possa girare praticamente senza modifiche sia in modo standalone che nel browser. La possibilità di sviluppare interfacce Omogenee sotto il minimo comune denominatore della tendenza al multimediale tipica di Flash

Directory: FLXB_2.0_Win_WWE.exe

JAVA SE DEVELOPMENT KIT 6

IL COMPILATORE INDISPENSABILE PER PROGRAMMARE IN JAVA

Se avete intenzione di iniziare a programmare in Java oppure siete già dei programmatori esperti avete bisogno sicuramente del compilatore e delle librerie Java indispensabili. Sotto il nome di Java SE Development Kit vanno appunto tutti gli strumenti e le librerie nonché le utility necessarie per programmare in JAVA. L'attuale versione è la 6.0, ovvero la nuovissima release densa di innovazioni e molto più legata al desktop di quanto non fossero tutte le precedenti

Directory: jdk-6-windows-i586

ECLIPSE SDK 3.2.2

L'IDE TUTTOFARE

Eclipse è un progetto completo portato avanti da Eclipse Foundation con la collaborazione di una miriade di aziende fra cui IBM, Adobe, Sun e che si è prefissata lo scopo di creare un IDE estendibile per plugin adattabile a qualunque tipo di linguaggio o tecnologia. Di default Eclipse si pro-

pone come IDE per Java ed è qui che da il meglio di se. Ma proprio grazie ai suoi plugin è possibile utilizzarlo come ambiente di programmazione per PHP, per C++, per Flex e per molti altri linguaggi ancora. Inoltre sempre grazie per ciascun linguaggio sono disponibili altri plugin ad esempio per rendere l'ambiente RAD o per favorire lo sviluppo dei Web Services o altro. Insomma lo scopo è stato raggiunto completamente. Eclipse è realmente un IDE tuttofare, ormai maturo, e che serve una miriade di programmatori grazie alle sue caratteristiche di affidabilità e flessibilità. Unica nota negativa: una certa pesantezza che lo rende idoneo ad essere usato solo su PC con una dotazione hardware minima di tutto rispetto

Directory: eclipse-SDK-3.2.1-win32.zip

PHP 5.2.1

IL LINGUAGGIO DI SCRIPTING PIÙ AMATO DEL WEB

Sono tre le colonne portanti di Internet: PHP, APACHE e MySQL. Certo la concorrenza è forte. Asp.NET e SQL Server avanzano con celerità, ma a tutt'oggi non si può affermare che i siti sviluppati in PHP costituiscano la stragrande maggioranza di Internet. Quali sono le ragioni del successo di cotanto linguaggio? Prima di tutto la completezza. PHP ha di base tutto quello che serve ad un buon programmatore, raramente è necessario ricorrere a librerie esterne, e quando è proprio indispensabile farlo esistono comunque una serie di repository che rendono tutto immediatamente disponibile ed in forma gratuita. Il secondo punto di forza del linguaggio sta nella sua capacità di poter essere utilizzato sia in modo procedurale che nella sua forma ad oggetti certamente più potente e completa. Esiste un terzo di punta di forza essenziale che è quello riguardante la curva di apprendimento. PHP è in assoluto uno dei linguaggi con la curva di apprendimento più bassa nel panorama degli strumenti di programmazione. Si tratta perciò di uno strumento indispensabile per chi si avvicina alla programmazione web, a meno che non intendiate scegliere strade diverse quali possono essere ASP.NET o JSP

Directory:php-5.2.0-Win32.zip

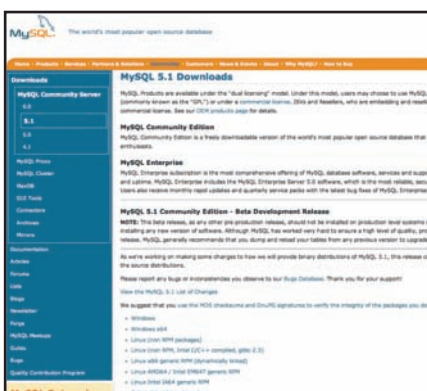
WORDPRESS 2.1.3

IL RE DEI BLOG

Wordpress è probabilmente lo strumento più diffuso per la programmazione di un vostro blog.Si tratta di uno strumento particolarmente leggero e semplice da usare.Nonostante questo l'architettura a plugin lo rende particolarmente estendibile. Molto interessante la struttura delle API che con una serie di hook rende la programmazione dei plugin un'attività relativamente semplice da svolgere

Directory: wordpress/wordpress-2.1.3.zip

MYSQL 5.1.15

IL PRINCIPE DEI DATABASE

Indispensabile per programmare webapplication in tecnologia PHP. Nonche non sia possibile utilizzare altridatabase, ma MySQL e PHP rappresentano veramente un binomioinscindibile. L'integrazione fra questo database e il linguaggio di scripting più usato sulla rete è talmente alta da fare divenire quasi un obbligo l'uso congiunto di questi due strumenti

Directory:mysql-5.0.27-win32.zip

PYTHON 2.5

L'EX GIOVANE RAMPANTE

Python è stato considerato per lungo tempo il nuovo che avanza. Attualmente non lo si può più definire in questo modo, Python è ormai un linguaggio stabile e completo che trova applicazione inun gran numero di progetti. Se ne parla sempre di più in campo industriale come su Internet. Soprattutto un gran numero di applicazioni anche in ambiente Windows girano ormai grazie a Python e presentano interfacce grafiche ottimamente strutturate. Ciò nonostante Python rimane un grande linguaggio di scripting adatto a gestire in modo completamente automatico buona parte di un sistema operativo sia esso Linux o Windows

Directory: python-2.5.msi

WXWIDGETS 2.8.4

COMODE LIBRERIE PER LO SVILUPPO DI INTERFACCE GRAFICHE

Interessantissime queste librerie, più volte le abbiamo utilizzate all'interno di ioProgrammo per realizzare degli esempi. Si tratta di librerie che consentono la creazione di interfacce grafiche, possono essere utilizzate da C++ ma anche da altri linguaggi come ad esempio Python. La cosa estremamente interessante è che consentono lo sviluppo di applicazioni completamente multipiattaforma, sono disponibili infatti sia in ambiente unix che in ambiente Windows.

Directory: wxMSW-2.8.4-Setup.exe

ALGORITMI DI SCHEDULING

LA VASTA CASISTICA DI PROBLEMI DI SCHEDULING HA IMPOSTO LA FORMULAZIONE DI NUMEROSI ALGORITMI. SI TRATTA DI SOLUZIONI DI OTTIMIZZAZIONE COMBINATORIA CHE SPESSO SONO FACILITATI DALL'USO DI GRAFI.

Assegnare delle risorse a richiedenti che devono svolgere lavori differenti cercando di minimizzare il tempo totale. È questo un modo per descrivere i problemi di programmazione dei lavori, più comunemente conosciuti come problemi di scheduling. Il compito gestito dai sistemi operativi multiprogrammati di assegnazione della CPU ai vari processi che ne fanno richiesta è il tipico e più popolare problema di scheduling, nell'ambito della programmazione. Ma i casi in cui sono applicabili algoritmi di scheduling sono davvero molteplici. La gestione di una catena di montaggio in cui un lavoro, come ad esempio l'assemblaggio di un auto, consta delle prestazioni in serie di più macchine. La coordinazione di servizi in un ufficio, come un'agenzia postale, dove si devono garantire le richieste dei cittadini a fronte delle attività degli impiegati cercando di minimizzare il tempo di attesa. Insomma, si tratta di problemi in cui si ha a che fare con due insiemi, quello delle risorse o macchine e quello delle attività o lavori che devono essere messi in relazione tra loro cercando di ottimizzare una funzione obiettivo che sicuramente dipende dal tempo. Abbiamo già classificato questi tipi di problemi dedicando ampio spazio ad esempi.
In questo numero focalizzeremo l'attenzione alle soluzioni, quindi esploreremo il maggior numero possibile di algoritmi. Prima di farlo fornirò in estrema sintesi tutte le informazioni utili per comprendere i riferimenti formali. Ovviamente, per ulteriori approfondimenti circa l'ambito di riferimento e alcune specifiche rimando al numero scorso di ioProgrammo, sempre nella sezione soluzioni.

PREMESSA

La teoria dello scheduling si fonda su una formulazione standard del problema che prevede macchine e job. Facilmente si possono associare le macchine e i job a qualsiasi tipo di risorse e attività un ipotetico problema preveda. Si hanno m macchine e n job che per espletarsi devono utilizzare macchine. L'obiettivo è completare tutti i job minimizzando o massimizzando, a seconda della formulazione, una funzione obiettivo che sicuramente terrà conto del tempo totale. Il servizio di una singola macchina per la realizzazione di un job è conosciuto come task. In alcuni casi un job viene completamente realizzato su un'unica macchina, come ad esempio nei sistemi multiprogrammati a singola CPU, così il job e il task coincidono. In figura 1 vi è una rappresentazione generica di problemi di scheduling.

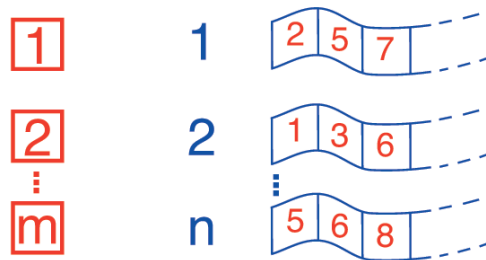



Fig. 1: "Problema di scheduling. Vengono messi in relazioni macchine in rosso e job in blu"

I tempi che descrivono il problema sono in sintesi:
Durata pij: tempo di processamento, è il tempo richiesto dalla macchina *i* per eseguire il task per il job *j*. Si tratta dell'i-esimo task del job *j*.
Release date rj: è il tempo di rilascio, rispetto al tempo iniziale *ti=0*, indica quando è possibile cominciare l'esecuzione del job.
Due date dj: è il tempo di consegna. Sempre rispetto al tempo iniziale *ti=0* indica entro che tempo bisogna terminare il job. Nella pratica il non rispetto di tale vincolo porta a penali o a rallentamenti dell'intero sistema.
Tempo di completamento Cj. È il tempo in cui termina il job *j*. I tempi sono riferiti al tempo iniziale *ti=0* di avvio del processo.
Lateness Lj. È la differenza tra il *tempo di completamento* e il *due date* ed è riferito al job *j*. Vale quindi *Lj=Cj-dj*. Esprime un ritardo se positivo, un anticipo se negativo.
Tardiness Tj. Questo indice è uguale alla Lateness nel

NOTA

CLASSIFICAZIONE

La principale classificazione dei problemi di scheduling prevede tre tipi:
A macchina singola: si ha una sola macchina, come per la gestione dei sistemi multiprogrammati a singola cpu. In questo caso job e task sono due termini che significano la stessa cosa.
Flow shop: è la catena di montaggio. Ogni singolo job per essere espletato deve passare nell'ordine per le m macchine.
Job shop: è il caso più generale. Ogni singolo job può usare un numero k, minore o uguale a m, di macchine è non necessariamente nello stesso ordine.

caso di ritardo, ossia se *Lj* è positiva, zero altrimenti. Tj=max{0, Lj}.

UN SEMPLICE ORDINAMENTO

Come è facile intuire la classe di problemi più facile da trattare è quella a macchina singola. Anche se le cose possono rapidamente complicarsi nelle situazioni più spinose in cui ad esempio è prevista la preempion o il set-up (vedi box). Analizziamo il caso in cui la funzione obiettivo sia una somma pesata del prodotto tra tempi rispetto alle priorità (peso) dei singoli job.

$$1, \quad , \sum_{j=1}^{n} w_j C_j$$

La soluzione si ottiene semplicemente ordinando i job secondo valori non decrescenti del rapporto tra tempi e pesi (*pj/wj*). È un modo semplice ed efficace per tener conto del tempo e del peso di ogni singolo job. Il metodo è conosciuto con la sigla *WSPT* - Weigheted Shortest Processing Time, ossia si elabora per primo il job che ha minore peso, in rapporto al tempo. E così si scelgono i successivi job. Si noti che se non sono previsti pesi, ossia tutti i valori *wi* sono 1, allora la soluzione consiste nel semplice ordinamento per tempo. Si dimostra che la soluzione proposta è ottima per il problema, così come formulato ad inizio paragrafo. Trattandosi di un ordinamento la complessità è quella associata agli algoritmi di sorting, quindi O(nlog(n)).

ALGORITMO DI LAWLER

Le cose cominciano a complicarsi quando la funzione obiettivo diventa più articolata, anche se molto più generale del caso precedente. Consideriamo la situazione in cui si voglia minimizzare una funzione obiettivo che prevede il massimo tempo di completamento espresso, non come semplice valore, ma come una funzione regolare. Si vuole in altri termini minimizzare una funzione obiettivo molto generale espressa come maxj{gj(Cj)}, con gj(Cj) funzione non decrescente di Cj detta appunto regolare.
Per ogni job c'è una funzione.
L'obiettivo è quello di minimizzare il valore più alto tra tutte le funzioni. Un esempio immediato di funzione g è il lateness ossia la differenza tra tempo di completamento e tempo di consegna. Ma si potrebbero fare a tale proposito molti altri esempi.
Nello sviluppare l'algoritmo solutore descriviamo in modo esaustivo il problema. Consideriamo il caso in cui sono presenti relazioni di precedenza. Così abbiamo il problema:

$$1, \textit{precedenza} \quad , \max_j \{g_j(C_j)\}$$

Sappiamo che questi tipi di problemi possono essere adeguatamente trattati usando dei grafi con i quali si rappresentano i vincoli di precedenza. A partire dal grafo che indica le precedenze, l'algoritmo di Lawler funziona a ritroso, si esaminano i nodi che non hanno archi uscenti; ossia quei job che non sono propedeutici a nessun altro lavoro. Tra questi se ne sceglie uno. Conoscendo il tempo di completamento, si sceglie tra i job dell'insieme quello che ha il minimo g rispetto al tempo di completamento. Si elimina il job e si itera il procedimento fino a svuotare l'intero grafo dai nodi. Si noti che a ogni cancellazione va aggiornato l'insieme dei job che non hanno archi uscenti. Se, infatti, si elimina un nodo che era terminale di un arco per un altro nodo (job), che a sua volta non ha altri archi uscenti, allora questo secondo nodo può essere aggiunto all'insieme citato. Con un teorema si dimostra che il problema esaminato ha una soluzione ottima applicando l'algoritmo di Lawler è che la complessità computazionale è $O(n^2)$.
Una sostanziale semplificazione si ha se la funzione g è ad esempio il *lateness*; si avrà il problema --1, , L_{max} - qui è più facile trovare il minimo della funzione g, è infatti quello con *due date* più grande tra i job non ancora sistemati (sequenziati), secondo la logica al ritroso descritta.
Ciò significa che è necessario ordinare i job per data di consegna in modo non decrescente. Questa seconda variante prende il nome di *EDD* (Earliest Due Date), e ha complessità analoga a *WSPT*, ossia O(nlog(n)).

QUALCOSA DI SIMILE ALLO SCHEDULING DI SISTEMA

Sempre nella tipologia della macchina singola rivestono particolare interesse i problemi preemptive, dove un job può essere interrotto.
Un esempio sono i sistemi operativi multiprogrammati dove bisogna assegnare la CPU, risorsa interrompibile, ai vari processi. Una formulazione di questa classe è *-1, rj | preemption, Lmax-*.
Nella ricerca operativa un primo approccio a questa famiglia di problemi prevede una modifica di EDD. Si considera l'insieme R(t)={j|rj<=t} che indica tutti i job rilasciati dopo un generico istante di tempo t. Se un job è in esecuzione necessariamente apparterrà a questo insieme. Con *PEDD* (Preemptive EDD) si decide che ad ogni generica iterazione del ciclo di scelta del job da eseguire si opti proprio per il job con tempo di consegna più basso tra quelli non terminati nell'insieme R(t). Anche qui un teorema, la cui dimostrazione come per i precedente casi è rimandata all'approfondimento del lettore, garantisce che tale algoritmo è una soluzione ottima per il problema appena

formulato, e che la sua complessità computazionale è dell'ordine di O(nlog(n)). La classe di problemi simile alla precedente dove non è consentito avere interruzioni: *-1, rj , Lmax-* è più difficile da trattare.
Si colloca nei problemi NP. È un caso di ottimizzazione combinatoria dove bisogna scegliere permutazioni di job, per farlo si usa il metodo del branch and bound. Si costruirà un albero che contiene in se le sequenze possibili.
La radice è un nodo che indica la sequenza vuota, i nodi figlio sono i possibili modi di iniziare la sequenza e così via. Così preso un generico nodo dell'albero ad esso corrisponderà una sequenza parziale. È applicando il metodo branch and bound che si risolve il problema.

PROBLEMI MULTIOBIETTIVO

Un'interessante classe di problemi, sempre per la categoria a macchina singola, prevede diversi obiettivi. In molti casi, infatti, bisogna perseguire più obiettivi che tra loro possono essere contrastanti. Diamo un cenno di come si possono affrontare tali tipi di problemi. Semplifichiamo e supponiamo che gli obiettivi siano due. Va riconsiderato il concetto di soluzione ottima visto che le funzioni obiettivo sono più di una. È provvidenziale il concetto di *nondominato*. Ad uno schedule S vengono applicate due funzioni obiettivo *f1(S)* e *f2(S)* da minimizzare. Un inciso, per le funzioni non è una perdita di generalità considerare la sola minimizzazione e non la massimizzazione, visto che con opportuni cambi di segno esse si possono sempre esprimere nella forma da minimizzare. S si dice non dominato se non esiste altro schedule *Sx* tale che *f1(Sx)* <= *f1(S)* e *f2(Sx)* <= *f2(S)*, dove una delle due disuguaglianze deve essere verificata in senso stretto, ossia per sola minoranza. Cosa indica che uno schedule S è nondominato? Che è possibile trovare uno schedule migliore solo per una funzione obiettivo, peggiorando però la situazione per l'altra delle funzioni obiettivo. È questa una situazione che va quindi trattata opportunamente. Si possono usare due metodologie diverse, con la prima si riformula il problema ad un singolo obiettivo ottimizzando rispetto a *f1* e ponendo la funzione *f2* come vincolo sul massimo valore che può assumere. Oppure viceversa. La seconda soluzione prevede di trattare tutte soluzioni nondominate.

FLOW SHOP

Terminata l'analisi di problemi a singola macchina concentriamoci su quelli con m macchine. Incominciamo con il caso del flow shop. Per questa categoria di problemi sono stati sviluppati una serie di metodi che si distinguono dai precedenti. Si suppone che siano presenti dei buffer tra una macchina e la successiva adeguati a mantenere i job in attesa di usare una determinata macchina. È come per una catena di montaggio, ad esempio quella automobilistica dove tra una macchina e l'altra siano previste delle aree di sosta che consentano di tenere l'auto in lavorazione in attesa che la macchina successiva termini il suo lavoro. Il buffer, inoltre, ha l'importante compito di risequenziare i job. Ovvero, i job presenti in un buffer non necessariamente devono essere trattati secondo una logica FIFO (first in first out) ma possono essere riordinati al fine di ottimizzare i lavori.
Un tale approccio aumenta il numero di soluzioni visto che le sequenze di job potendosi riorganizzare sul buffer di una generica macchina aumenta. In particolare si ha *(n!)^m*. Questi sorpassi sono in effetti una variante del menzionato flow shop che diventa *permutation flow shop*.
Per il classico flow shop il sequenziamento è unico e genera una sola permutazione quindi le possibili soluzioni sono *n!*. Anche in questo caso si possono usare molti algoritmi a seconda della specifica del problema. Per capire considereremo il problema di makespan indicato come *-Fm, , Cmax-*. Con *m* il numero di macchine. Introduciamo una struttura che sia in grado di modellare le variabili e le funzioni del problema e che può essere usata anche per altri problemi. Si tratta di un grafo che associa un nodo ad ogni possibile task. Questi possono essere m per ogni job quindi *mn*. Per la precisione il grafo consta di nm+2 nodi. Per capirci il nodo [i,j] rappresenta il task i del job *j*.
Ad ogni nodo si può associare il peso *pij*. Anche qui quindi si possono esprimere i vincoli di precedenza orientando il grafo, come descritto prima e più approfonditamente nello scorso articolo. Gli archi così introdotti si dicono orizzontali. Il grafo tiene conto anche del sequenziamento di task sulle singole macchine che saranno espressi da archi non orientati. Tali archi sono detti disgiuntivi. Infine, va aggiunto un nodo sorgente che quindi non ha archi entranti e un nodo pozzo senza archi uscenti. I successori del sorgente sono i nodi [1,j], mentre predecessori del pozzo sono [m,j].
Si tratta di una struttura complessa. La si può vedere come un grafo che contiene al suo interno altri grafi. Se, infatti, si considera una generica macchina *i* per essa si può considerare il sequenziamento di job che possiamo indicare come *li*. L'insieme di tutti gli *m* sequenziamenti sarà *l*.
Se G è il grafo completo prima descritto, con *G(l)* si indica il grafo orientato ottenuto applicando i sequenziamenti. In realtà è proprio *G(l)* il grafo delle attività descritto nello scorso articolo, con i nodi che corrispondono ai task e gli archi alle precedenze.
Vi è un importante risultato dimostrabile. Dato *l* un insieme di sequenziamenti, il makespan è dato dal peso del cammino minimo di peso massimo dal nodo sorgente al nodo pozzo su *G(l)*. Il quesito si sposta

VINCOLI
Vi possono essere diverse caratteristiche associate ad un problema di scheduling. Le più note sono:
Vincoli di precedenza: alcuni task per essere eseguiti necessariamente devo attendere che altri terminino.
Preemptive: è il caso in cui è possibile interrompere un task.
Set-up sono dei tempi previsti per l'avvio di alcune macchine

quindi su un consolidato problema della ricerca operativa applicata ai grafi aciclici, in particolare quello per la ricerca di un percorso massimo.
È quindi si tratta di trovare uno schedule, ossia un insieme di sequenziamenti per le varie macchine che minimizzi il makespan (massimo tempo di completamento).

ALGORITMO JOHNSON

Per risolvere il problema consideriamo un semplice caso in cui vi sono soltanto due macchine. Una soluzione storica fu data da Selmer Johnson che sviluppo l'algoritmo omonimo.
Si suppone che una delle due macchine sia sempre attiva. Poiché si vuole minimizzare il tempo di completamento si dovrà fare in modo da massimizzare il tempo di utilizzo della seconda macchina.
Per far questo Johnson pensò che il buffer tra la prima e la seconda macchina si doveva tenere il più possibile pieno al fine di alimentare di "continuo" o almeno il più possibile la seconda macchina.
Un modo di procedere per favorire questa situazione è dare alla prima macchina un job che sia il più breve possibile in modo che termini subito, e che quindi si possa rapidamente dare da fare alla seconda macchina.
Allo stesso tempo è conveniente che la macchina due sia sottoposta a job lunghi in modo da far riempire il più possibile il buffer. E si prosegue così alimentando man mano di job brevi, ma comunque di tempo sempre maggiore la macchina 1; e job di lunga durata ma di tempi via via minori la macchina due.
Questo modo di procedere porta ad una soluzione ottima.

JOB SHOP

Facciamo sempre riferimento al problema di makespan.
Il job shop viene trattato in modo analogo al flow shop, bisogna però tener conto di altre informazioni.
Per ogni task non è più sufficiente conoscere *pij*, ossia il tempo di esecuzione del task *i* del job *j*, ma sarà necessario conoscere *mij* che individua la macchina dove quel task deve essere svolto.
Ovviamente, questa informazione non serviva nel caso di flow shop nella sua formulazione standard. Quindi per ogni job *j* si avrà una sequenza di macchine, in generale diversa da job a job; questa combinazione è conosciuta come instradamento.
Il grafo G per la descrizione del problema è sostanzialmente simile al grafo costruito nel caso precedente dove i nodi rappresentano i task e gli archi le precedenze.
Bisogna aggiungere gli archi disgiuntivi, prima accennati, per specificare i task della stessa macchina. Non saranno necessari archi che per identico job visto che questa informazione si può dedurre dai vincoli di precedenza e dagli archi ad essi associati. Detto *li* un sequenziamento di task su una macchina e *l* un insieme di sequenziamenti allora è possibile ottenere dei risultati. Se nel grafo *G(l)* si presentano dei cicli allora il sequenziamento è inammissibile.
Se, invece, il grafo *G(l)* è aciclico, il makespan è dato dal peso del cammino di peso massimo dal nodo sorgente a quello destinazione. In questo caso quindi si pone un problema aggiuntivo che è quello di verificare che lo schedule sia ammissibile.
Risulta evidente che trattare con problemi di questo tipo non è facile, basti pensare che il grafo contiene *mn+2* nodi che in situazioni medio grandi possono produrre strutture davvero "consistenti", a ciò si deve aggiungere che il numero di archi è ancora più elevato visto che ve ne sono di due tipi orizzontali e disgiuntivi. Insomma, si tratta di problemi che richiedono soluzioni mirate.
Una soluzione alla formulazione appena fatta è un euristica conosciuta come shifting bottleneck. Questa è associata al problema *-J,,Cmax-*, che nel campo dell'ottimizzazione combinatoria è uno dei casi più difficili. Senza entrare nei particolari verrà esposta l'idea di fondo su cui è stato sviluppato l'algoritmo. Si deve trovare un insieme di sequenziazioni *l* tale che il grafo *G(l)* sia aciclico e che la lunghezza del percorso critico su tale grafo sia la più bassa possibile. Tra le varie macchine ci può essere una che per così dire è più "critica", ossia che funga da collo di bottiglia (bottelneck). Si pensi alla semplice schedulazione di lavori d'ufficio, spesso si assiste a file davanti macchine particolarmente lente come ad esempio la stampante. Per questo sarebbe innanzitutto necessario dimensionare bene alcuni aspetti del sistema, come per l'esempio il numero di stampanti e la loro velocità.
Ma qui sottolineiamo l'importanza di usare un metodo adatto. L'algoritmo si preoccupa di trovare i sequenziamenti "migliori" proprio per i colli di bottiglia, in modo da velocizzare l'intera attività.

CONCLUSIONI

In questo articolo ci siamo lanciati nella presentazione di un gran numero di algoritmi. Tipicamente ciascuno di questi algoritmi potrebbe seguire una propria strada per quantità di argomenti correlati e tipologia di approccio.

Fabio Grimaldi

NOTA

RAPPRESENTARE LO SCHEDULING

Bisogna specificare tre valori separati da virgola. Il primo dei valori può valere 1, F o J e indica rispettivamente scheduling a macchina singola, flow shop e job shop. Il secondo descrive alcune caratteristiche più specifiche come vincoli di precedenza, set-up, preemption e così via. Infine l'ultimo dei parametri è la funzione obiettivo. Ad esempio un sistema a singola macchina preemptive che tenga conto dei tempi di esecuzione e dei pesi di ogni job può essere descritta come segue:

1, preemption,

$$\sum_{j=1}^{n} w_j C_j$$

via via minori la macchina due.
Questo modo di procedere porta ad una soluzione ottima.

JOB SHOP

Facciamo sempre riferimento al problema di makespan.
Il job shop viene trattato in modo analogo al flow shop, bisogna però tener conto di altre informazioni.
Per ogni task non è più sufficiente conoscere *pij*, ossia il tempo di esecuzione del task *i* del job *j*, ma sarà necessario conoscere *mij* che individua la macchina dove quel task deve essere svolto.
Ovviamente, questa informazione non serviva nel caso di flow shop nella sua formulazione standard.
Quindi per ogni job *j* si avrà una sequenza di macchine, in generale diversa da job a job; questa combinazione è conosciuta come instradamento.
Il grafo G per la descrizione del problema è sostanzialmente simile al grafo costruito nel caso precedente dove i nodi rappresentano i task e gli archi le precedenze.
Bisogna aggiungere gli archi disgiuntivi, prima accennati, per specificare i task della stessa macchina.
Non saranno necessari archi che per identico job visto che questa informazione si può dedurre dai vincoli di precedenza e dagli archi ad essi associati.
Detto *li* un sequenziamento di task su una macchina e *l* un insieme di sequenziamenti allora è possibile ottenere dei risultati. Se nel grafo *G(l)* si presentano dei cicli allora il sequenziamento è inammissibile.
Se, invece, il grafo *G(l)* è aciclico, il makespan è dato dal peso del cammino di peso massimo dal nodo sorgente a quello destinazione. In questo caso quindi si pone un problema aggiuntivo che è quello di verificare che lo schedule sia ammissibile.
Risulta evidente che trattare con problemi di questo tipo non è facile, basti pensare che il grafo contiene *mn+2* nodi che in situazioni medio grandi possono produrre strutture davvero "consistenti", a ciò si deve aggiungere che il numero di archi è ancora più elevato visto che ve ne sono di due tipi orizzontali e disgiuntivi. Insomma, si tratta di problemi che richiedono soluzioni mirate.
Una soluzione alla formulazione appena fatta è un euristica conosciuta come shifting bottleneck.
Questa è associata al problema *-J,,Cmax-*, che nel campo dell'ottimizzazione combinatoria è uno dei casi più difficili. Senza entrare nei particolari verrà esposta l'idea di fondo su cui è stato sviluppato l'algoritmo.
Si deve trovare un insieme di sequenziazioni *l* tale che il grafo *G(l)* sia aciclico e che la lunghezza del percorso critico su tale grafo sia la più bassa possibile.
Tra le varie macchine ci può essere una che per così dire è più "critica", ossia che funga da collo di bottiglia (bottelneck). Si pensi alla semplice schedulazione di lavori d'ufficio, spesso si assiste a file davanti macchine particolarmente lente come ad esempio la stampante. Per questo sarebbe innanzitutto necessario dimensionare bene alcuni aspetti del sistema, come per l'esempio il numero di stampanti e la loro velocità.
Ma qui sottolineiamo l'importanza di usare un metodo adatto. L'algoritmo si preoccupa di trovare i sequenziamenti "migliori" proprio per i colli di bottiglia, in modo da velocizzare l'intera attività.

CONCLUSIONI

In questo articolo ci siamo lanciati nella presentazione di un gran numero di algoritmi. Tipicamente ciascuno di questi algoritmi potrebbe seguire una propria strada per quantità di argomenti correlati e tipologia di approccio.
Abbiamo qui preferito trattare un gran numero di soluzioni piuttosto che approcciarne una in particolare.
Questo perché ciascun algoritmo si presta a risolvere in un modo determinato solo alcune problemi, mentre altri possono essere affrontati in modo diverso.
Una carrellata generale sui vari possibili approcci al problema dello scheduling consente di avere una visione globale delle possibili soluzioni e questo garantisce una maggiore possibilità di scelta.
Certamente qualunque algoritmo sceglierete per risolvere i vostri problemi di scheduling, avrere necessità di approfondire almeno qualcuna delle tematiche introdotte.
In più di un caso troverete anche utile mescolare uno o più degli algoritmi che qui abbiamo semplicemente presentato. In ogni caso il problema dello scheduling dei processi rimane affascinante e trova applicazioni non solo in informatica ma anche nel campo della gestione aziendale e della programmazione di attività

Fabio Grimaldi